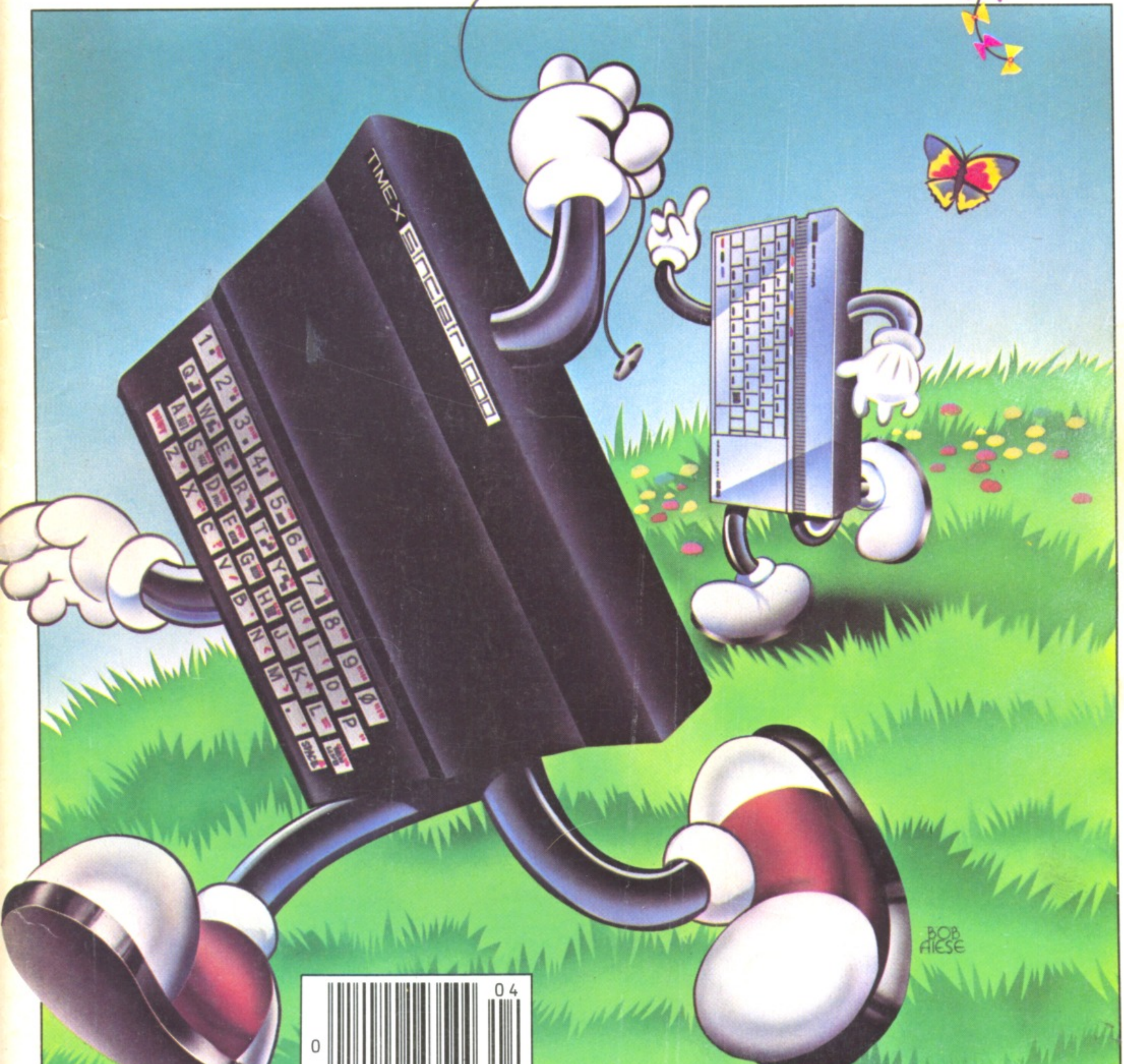
The magazine for Sinclair users and Timex/Sinclair users

$2.95

SYNC®

March/April 1984

Volume 4, Number 2



0 14024 14354 04

THE GAME LORDS OF BRITAIN

COMMODORE 64™

PURPLE TURTLES
Authors: **Mark & Richard Moore**
A fully animated arcade game with Loveable Turtles, Cuddly Graphics and more Cuteness than you'll find in any other Commodore 64 game! A game for the young at heart and people who have tired of alien bashing.

ULTISYNTH
Author: **Nalin Sharma**
Turn your CBM 64 into a sophisticated synthesiser, a piano, violin, organ, guitar or harpsichord. Woodwind, drums or cymbals and play along with your own or any of the preset rhythms.

Also available
Bugaboo • Aquaplane • Ring of Power • Sting • Quintic Warrior

TIMEX/SINCLAIR 2068™

BUGABOO
Author: **Indescomp**
Jump your way out of the caves with Bugaboo the flea but beware of the fearsome Dragon as you jump around the exotic vegetation.

Also available
Xadom • Astro Blaster • Games Designer • 3D Strategy • Ant Attack • The Chess Player • Smugglers Cove • Velnor's Lair • Trader • Traxx • Time Gate

TIMEX/SINCLAIR 1000 & 1500™

3D BLACK STAR
Author: **M. Sudworth**
Explosions mushroom around you and the rear scanner shows following craft, you increase speed and fire at the ducts, dodging from side to side of the narrow corridor.

Also available
Damper & Glooper • Croaka Crawla • Pioneer Trail • Munchees

ATARI 400/800™

ULTI-SKETCH
Author: **M. Walker**
INCREDIBLY VERSATILE CHARACTER GENERATOR
Define characters on 8 × 8 grid. Save to tape. Load from tape. Freely merges character sets into existing programs. Hex & Dec. O/Ps. Character manipulation. Mirror, rotate, invert, left, right, up down, hold, wipe, cancel. Redefine whole character set. Also works in colour mode. Excellent examples supplied.

VIC 20™

SKYHAWK
Author: **Chartec**
A quiet European village is attacked, pilot the jet fighter Skyhawk against the attackers.

Also available
Tornado • Bugaboo

QS™ THE GAME LORDS OF BRITAIN

QUICKSILVA INC. 426 West Nakoma San Antonio Texas 78216. (512) 340 3684.

CIRCLE 34 ON READER SERVICE CARD

MEMOTECH

The Complete Range

Fifteen months ago Memotech developed the first 64K Memopak, designed to maximise the capabilities of the Sinclair ZX81. Since then, using the ZX81 as a starting point, we've gone on to produce a comprehensive range of Memopaks, adding 16K and 32K memory expansions, utilities packages comprising a Word Processor, Z80 Assembler and Spreadsheet Analysis, plus Communication Interfaces, High Resolution Graphics and a professional quality Keyboard. To complete our range of Timex add-ons, we are now introducing the MEMOPAK RS232 Serial Interface.

RS232 Interface

The RS232 is an all-purpose interface which allows the Timex not only to output to suitable serial printers, but can link up with numerous types of peripheral or even other processors. The Interface has two main modes of operation: BASIC mode allows you to use the range of functions supplied in the RS232 EPROM within an ordinary BASIC program, and TERMINAL mode allows you to use your Timex as a terminal to another processor.
The EPROM functions offered permit the user to send, receive and convert bytes between Z80 code and ASCII, as well as check the status of numerous control flags. Received or transmitted data can appear simultaneously on the screen, and received data may be printed simultaneously.

$79.95 cable $19.95

Memopak Centronics I/F

The BASIC commands LPRINT, LLIST and COPY are used to print on any CENTRONICS type printer. All ASCII characters are generated and translation takes place automatically within the pack. Reverse capitals give lower case. Additional facilities allow high resolution printing.

$59.95 cable $19.95

Memopak HRG

This pack breaks down the constraints imposed by operating at the Z80 character level and allows high definition displays to be generated. All 248 x 192 individual pixels can be controlled using simple commands, and the built in software enables the user to work interactively at the dot, line, character, block and page levels.

$79.95

Memocalc

The screen display behaves as a 'window' on a large sheet of paper on which a table of numbers is laid out. The maximum size of the table is determined by the memory capacity, and with a Memopak 64K a table of up to 7000 numbers with up to 250 rows or 99 columns can be specified.

$39.95

Memotext

Text is first arranged in 32 character lines for the screen with comprehensive editing facilities. On output the user simply chooses the line length required for printing and the system does the rest. Used with the Memopak Centronics Interface, the Word Processor makes available printout with 80 character lines, upper and lower case and single and double size characters.

$39.95

Memopak Memory Extensions

For those just setting out on the road to real computing, these packs transform the Timex from a toy to a powerful computer. Data storage, extended programming and complex displays all become feasible. Further details available on request.

16K Memopak $39.95
32K Memopak $79.95
64K Memopak $119.95

Z80 Assembler

The Assembler allows you first to code and edit a source program in the Z80 language, and then assemble it into machine code. You can now write flexible and economic programs. The Editor mode allows you to code directly in the right format, manipulatė inidividual lines and control the exact placing of source and machine code. Routines may be merged or listed (even to a commercial printer using our Centronics Interface). The assembler mode handles all standard Z80 mnemonics, numbers in hex or decimal, comments and user-selected lables.

$39.95

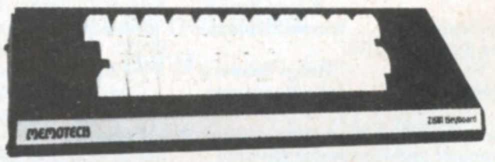
Memotech Keyboard

The Memotech plug-in Keyboard plus buffer pack takes the effort out of data entry for Timex users. The Keyboard has a light professional touch and is housed in an elegant aluminum case. The simple plug-in system means that you are not obliged to open up your Timex, use a soldering iron or invalidate your Timex warranty.

Keyboard Buffer Pak

The Buffer Pak performs a "housekeeping" function for the Keyboard, interfacing directly with the port of your Timex.

$79.95 – (keyboard & buffer included)



Note! All Memotech products carry a 6 mo. warranty. 80 column dot matrix printer packages available at a substantial savings from Memotech.

Order at no risk (10 day money-back guarantee): Call 617-449-6614. Or send your name, address, phone number and a check/money order / Visa or MasterCard number with expiration date to:
Memotech Direct Sales Division, 99 Cabot Street, Needham, MA. 02194.
Shipping/Handling $4.95; Massachusetts Residents add applicable sales tax.

The magazine for Sinclair users and TIMEX/Sinclair users

SYNC®

March/April 1984

Volume 4, Number 2

DEPARTMENTS

SYNC AT THE KEYBOARD

TS2068

MACHINE LANGUAGE

HARDWARE ARTICLES

ARTICLES

GAMES

REVIEWS

Staff

Volume 4, Number 2

SYNC (USPS: 585-490; ISSN: 0279-5701) is published bi-monthly by Ahl Computing, Inc., a subsidiary of Ziff-Davis Publishing Company. David Ahl, President; Elizabeth B. Staples, Vice-President; Selwyn Taubman, Treasurer; Bertram A. Abrams, Secretary. 39 E. Hanover Ave., Morris Plains, NJ 07950. Second class postage paid at New York, NY 10001, and at additional mailing offices.

Subscription rates: USA: One year (6 issues), $19.97; two years (12 issues), $36.97; three years (18 issues), $49.97. Canada and other foreign: $4 per year additional.

Subscriptions: For all inquiries concerning subscriptions, new orders, renewals, change of address (include the name and old ZIP or mailing label), problems, etc., write to: SYNC Magazine, PO Box 2939, Boulder, CO 80302.

Advertising: For advertising information, contact Wayne Stephens, *SYNC* Advertising Sales Manager, Ziff-Davis Publishing Company, One Park Ave., New York, NY 10016 (phone: 212/725-4216).

All other correspondence: Address to: *SYNC*, 39 E. Hanover Ave., Morris Plains, NJ 07950. In U.K. to: *SYNC*, 10 Bishops Way, Sutton Coldfield, W. Midlands B74 4XU.

Postmaster: Send address changes to: *SYNC*, PO Box 2939, Boulder, CO 80302.

Cover by **Bob Aiese**

503-5300

NEW MD-2B

MODEM

You can find a **Cheaper** Modem, But do you want a **Cheaper** Modem?
No other Modem offers the Smart software & features of the New BYTE-BACK MD-2B Modem.

2 FREE HRS. ON COMPUSERVE

SMART
MODEM

$119.95 KIT

ASSEMBLED & TESTED $149.95

With New **SMART**, Menu Driven SOFTWARE included FREE.

- Send and receive Programs by Phone
- Copy Information into memory from a Host Computer (such as **Compuserve**), Print it, Review it, Save it on Tape, Send from Memory
- **Operate in Originate or Answer Mode**
- Use Timex 2040 Printer or Any RS-232 Printer
- RS-232 Printer Port provided.
- No extra memory Required, But with 64-K Memory You Can Store Up To 60 Full Screens
- Adaptable to the 2068

COMPUSERVE PACKAGE WITH 5 FREE HOURS ONLY $39.95

BYTE-BACK modules

64-K MEMORY

With **Battery Backup** **$109.95** KIT

$119.95 ASSEMBLED & TESTED

the "ULTIMATE MEMORY" UM-64

FEATURES:

- **Battery Backup - Computer and full 64-K Backed**
- **PROM/ROM socket**
- **Reset Switch-Reset without destroying Program**
- **BYTE-BACK EXCLUSIVE FULL 64-K**

The 0-8K area is available. You can execute a copy routine (provided) to copy the TIMEX ROM into the 0-8K area of RAM then flip a switch and you have your operating system in RAM. You can modify it and create your own customized operating system. Full details, examples & programs included.

WHY PAY MORE FOR LESS FEATURES?
***GET THE* "ULTIMATE MEMORY"**
BYTE BACK'S UM-64

NEW ZX PRO/FILE

Cassette & Book **$16.95** including shipping

Clearly the Best File Management Program Available.
This New Program is what the Timex has been waiting for.

At **BYTE-BACK** we have used & evaluated almost every data organizer & Data File Program on the market & we were not really impressed until we tried **Thomas Woods' NEW ZX PRO/FILE** Program. It is truly the best piece of software we have seen for the Timex Computer. We now use it to help run our business. This program is so FAST that even when the program is full the file you want is on the screen within a second.

The ZX PRO/FILE has features not found on the other Data File programs. It has: **FILE ANALYSIS** function which allows you to count, analyze frequency, %, etc. Capability to create **FILES OF ANY SIZE** in the same program. **MULTIPLE WORD SEARCH,** excellent **ADD/EDIT** features. **ORDERED FILE OUTPUT** based on any numerical value contained in the files. **DEFINABLE PRINTER FUNCTIONS & AUTO SEARCH.** Works with both **16K & 64K** memory.

FREE 59 page **BOOK.** This book alone is worth the purchase price. ZX PRO/FILE's machine language concepts & methods are fully explained. It includes a complete **PROGRAM LISTING &** detailed explanation of how the program works & how it can be easily modified. It also includes a machine language programming section.

BYTE-BACK'S BB-1
CONTROL MODULE

$59.00 KIT *In Stock!*

ASSEMBLED & TESTED $69.00

ADAPTABLE TO THE 2068

- **8 Independent Relays** *(with LED status indicators)*
- **8 Independent TTL Inputs** *(with Schmitt trigger buffers)*

- By using a single POKE command you can change and latch the status of each of the 8 relays.
- Your computer can read the status of all 8 inputs by the use of a single PEEK command.
- A comprehensive manual is included that has complete application details.
- More than one BB-1 can be used at a time.

M-64 64K MEMORY $79.95 Close Out Special

We have discontinued our popular M-64 Memory & are closing out the few remaining units in stock. This is not a kit & has full warranty. It doesn't have battery back-up & a PROM socket like our new UM-64, but it is compatable with the 2040 printer, TS1000 & TS1500.

RS-232 Module $59.95

ASSEMBLED & TESTED **$69.95**

Drives the Gorilla/Bannana & Other Serial Printers

ALL MODULES CARRY 90-DAY WARRANTY

TRY BYTE-BACK MODULES FOR 10 DAYS WITH NO OBLIGATION

Gorilla/Banana Printer

Prints on Standard Paper. 80 Columns
Specify Serial or Parrallel.

Discounted to only **$239.95**

See "Leading Edge" Ads

BYTE-BACK
KEYBOARD **$39.95** without case

We use the same key switch found in the expensive computers such as: Hewlett Packard, Texas Instruments, Atari, D.E.C. etc. The multi-colored key caps are the same type used on the IBM Personal Computer, but with the TIMEX ledgens.

EXPERIENCE QUALITY

Shipping and Handling $4.95 EXCEPT ZX PRO/FILE

ORDER PHONE (803) 532-5812

ITEMS ORDERED: ______________________

Bill My ☐ Am. Exp. ☐ Visa ☐ M/C
Checks Accepted

Exp. Date ________ Card No. ______________

Name ______________________

Address ______________________

City/State/Zip ______________________

Phone ______________________

Dealer Discounts Available

Mail To: **BYTE-BACK CO.**
Rt. 3, Box 147, Brodie Road
Leesville, S.C. 29070

CIRCLE 10 ON READER SERVICE CARD

All products are compatible with TS1000 and TS1500 Computers

letters

64K RAM

Dear Editor:

I enjoy reading your magazine because of its great insight into the needs of its readers. You do a great job of providing information on the many uses and expansion possibilities of the TS1000, and you have done a marvelous job of giving inspiring suggestions for getting the most out of our simple yet powerful 2K computers. Some of your programs for those with 16K of power are really out of this world.

Yet I am at a loss to know how to feed my hungry 64K of computing power, and you can be sure I am not alone in this search for power. Many of us out here feel the TS1000 is as good as any other computer on the market, but we would like to know what we can do with our 64K RAM packs to make our computers operate more effectively.

George Bilokonsky
4002 Elaine Pl. S.
Columbus, OH 43227

"Using RAM Packs" in this issue is a start, and we would welcome other articles on the topic.

The ZTX-750 Transistor

Dear Editor:

I believe I have an answer for J. L. Peeler (*SYNC* 4:1). The equivalent transistor for the ZTX-750 or ZTX-752 may be Radio Shack's RS-276-2032.

Stephen Wilson
221 Forest Dr.
Linwood, NJ 08221

Dear Editor:

I was able to purchase the Ferranti ZTX-750 PNP transistor (used in the voltage converter section of the 16K RAM schematic in *SYNC* 1:5) from Myco Vickers, 8040-3 Deering Ave., Canoga Park, CA 91304 (213/340-2043). Also I switched the BA221 rectifier with an 1R30S1.

Better yet, as suggested by David Ornstein, avoid the voltage converter altogether by using Intel 2118 5V RAMs (if you can find them; they are no longer being manufactured).

Spuh Poyn
14441 Nordhoff St.
Van Nuys, CA 91402

Say What

Dear Editor:

In "Say What" by Brad Bennett (*SYNC* 3:6) the seven lines 800-830 (the CLEAR FILES routine) can be replaced by one line 800 RUN. This is all that is needed to clear the variables. Not only will this save programming space, but the RUN command is much faster than the Basic routine. In place of the audio amplifier, I found that the tape recorder hookup described by Ron LeMon in "AUDISY" in the same issue works fine for either program.

Gregory T. Blocker
PO Box 1226
Page, AZ 86040

TS2068

Dear Editor:

WOW!

That's my reaction to the TS2068.

However, a prediction: the TS1000 and 1500 are not dead, by any means. Those of us who are into computing tend to forget the mass market is looking for a feature/price ratio they like—and maybe they do not care about color and sound.

I intend to continue using my TS1000, though I have just gotten a TS2068, and will probably buy a TS1500 for developing programs for it and the TS1000; my kids will, I am sure, use the TS1000.

The conversion of one of my programs to run on the Apple IIe took more than 20 hours. The transfer to the TS2068 took four hours. Part of the difference is due to the need to store data on disks with the Apple, but part of it is due to the letter-by-letter entry of keywords, the lack of syntax checking on line entry, the loss of program data when a syntax error is encountered during a test run, and the need for extra symbols and letters such as LEN (A$) instead of the TS LEN A$. I have used the IBM PC, too, and it is not much better (if it is at all better) in this respect. Let's hear it for Timex Sinclair Basic!

SYNC has far and away the most and best material for TS users. The general interest computer magazines tend to ignore it—their mistake, I think.

John G. Sandell
2 E. Oak Ave.
Moorestown, NJ 08057

Bulletin Boards

Dear Editor:

Are there any free bulletin boards out there that can be accessed by a modem such as the Byte-Back MD-1? Perhaps *SYNC* should publish of list of them including the phone number, the city location, the SYSOP name, and whether the board has uploading/downloading program capabilities.

Joseph Lavinus
1911 N. Van Buren St.
Arlington, VA 22205

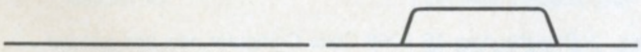
We will publish information on bulletin boards and services accessible by modems in our Resources Column. Full details should be sent to: "Resources."

A NEW, IMPROVED TS1000/ZX81 KEYBOARD AT A NEW, IMPROVED PRICE. $15.

There's only one thing wrong with the TS1000/ZX81. Its keyboard.

Or rather its lack of one.

Since it's flat your fingers don't feel as if there's any response to the pressure put on the keys.



TS1000/ZX81 KEYS SECONDBASE KEY

In other words, you're not quite sure which keys you've pressed until the screen actually tells you.

Our new, improved push button keyboard changes all that.

It matches the TS1000/ZX81 perfectly. And the keys give a real calculator-type feel.

To set it up all you have to do is peel off the adhesive backing and stick it on top of the TS1000/ZX81 touchpad.

Because no tampering or soldering is involved the guarantee is not affected. And it will last for up to 3½ million operations.

But our keyboard doesn't just come loaded with features. With it comes a separate overlay and a set of coloured stick-on labels to make game playing easier.

It's yours for the new, bargain price of $15.

Whichever way you look at it, we think you'll agree that it's a keyboard that's quite outstanding.

Please send______Keyboards at $15.00 each.
Enclosed is my check or money order for $______ (US Dollars only)

Please charge my Visa ☐ Mastercard ☐

Account # ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ ☐☐ ☐☐ Expiry date

Name

Address

City/State/Zip

Telephone

Signature

SECONDBASE

700 Lexington Avenue, PA 16601 U.S.A. Tel: 814-942 8171.

Read This First

Before you enter the programs in this issue, please note:

All the programs require the **TS1000** or **ZX81** with **16K RAM** unless other requirements are given at the top of the first page of the article. Listings with lower case letters are for the **TS2068**.

Read the article all the way through before trying to enter the program.

A letter after a number shows the type: b for binary; d for decimal; h for hexadecimal.

In PRINT statements:

#: Enter a necessary space.

A (32): The underline means use the graphic on that key. The number in ( ) tells how many times.

A̅: The overline means use the key in inverse.

INPUT: An underlined *word* found on the keyboard should not be spelled out. Enter it directly. If it will not ENTER, hit THEN, then the keyword you want, backspace, delete THEN, and continue entering the line. This memory saving technique may be disregarded if you have enough RAM.

glitchoidz report

Search and Replace (4:1)

Version I, Routine A, line 9: Replace inv. Y with quotation marks.

Version II, Routine B, line 1: Change LD HL,16566 to LD HL,16564; replace inv. Q with inv. O (letter).

Brick Busters (4:1)

P. 4, col. 3.

Make the following change in the list of numbers: 42,12,64,35,54,*135*,. . .

Tax Shelter Time Bomb (4:1)

40 IF Y<1 OR Y>=13 THEN . . .

try this

TS1000/ZX81 (1K RAM or more)

Type in the following lines. Make sure the computer is in SLOW mode, press RUN and ENTER. Type in any character or string of characters and press ENTER. Notice the cursor response. When the screen is full, hit CONT, ENTER, and continue typing.

```
10 POKE 16390,INT (RND*256)
20 INPUT A$
30 PRINT A$
40 GOTO 10
```

TS2068

Change the 16390 in line 10 to 23617. Notice the difference from the TS1000.

```
10 POKE 23617,INT (RND*256)
20 INPUT A$
30 PRINT A$
40 GOTO 10
```

Our thanks to:
Christopher D. Marsh
911 Summit Dr.
Greensburg, PA 15601

"Try This" features short programs to show off your computer, impress your family and friends, and tickle your imagination when *SYNC* arrives at your place. Send your contributions to: Try This, SYNC, 39 E. Hanover Ave., Morris Plains, NJ 07950.

TS1000/ZX81 (2K RAM)

Type in the lines below, press RUN and ENTER. The BREAK key will stop the program at any value of X (degrees).

```
  5 FAST
 10 FOR A=0 TO 63
 20 PLOT A,21
 30 NEXT A
 40 FOR B=0 TO 43
 50 PLOT 0,B
 60 NEXT B
 65 SLOW
 70 FOR X=0 TO 62
 80 PRINT AT 18,10;"Y= 20 SINE
X"
 85 PRINT AT 20,10;"X=";X*360/6
2
 90 LET Y=20+20*SIN (X/31*PI)
 95 PRINT AT 21,10;"Y=";(Y-20)
100 PLOT X,Y
110 NEXT X
```

TS2068

Delete lines 5 and 65 in the above listing and type in. Notice the difference from the TS1000.

Our thanks to:

Ted A. Pozyski
1215 Thompson St.
Houston, TX 77007

TS1000/ZX81 (1K RAM)

Carefully examine the lines below. Decide what the computer should PRINT. Now type in the lines:

```
  1 REM BOVEE
 10 IF .02+.03=.05 THEN PRINT "
THEY ARE EQUAL"
 20 PRINT .02+.03
```

Press RUN and ENTER. Did you get what you expected? Our thanks to:

Chris BoVee
Box 8264
Pembroke Pines, FL 33084

TS2068

Type in the lines below, RUN, and

```
 10 FOR d=95 TO 255 STEP 10
 15 PLOT 65,27: DRAW 100,100,PI
*d
 20 PAUSE 30: CLS : NEXT d
```

```
 10 OVER 1
 15 FOR n=1 TO 2
 20 PLOT 65,27: DRAW 100,100,PI
*255
 25 NEXT n
```

ENTER. Observe the results. Our thanks to:

Sharon Zardetto Aker
20 Courtland Dr.
Sussex, NJ 07461

TS1000/ZX81 (2K RAM)

Type in the lines below. RUN in SLOW

```
 10 FOR A=0 TO 10
 20 FOR C=A TO 31-A
 30 PRINT AT A,C;"▀"
 40 IF C>(29-A) THEN GOTO 60
 50 PRINT AT 21-A,C+1;"▄"
 60 NEXT C
 70 FOR L=A+1 TO 20-A
 80 PRINT AT L,31-A;"▌";AT 21-L
,A;"▐"
 90 NEXT L
100 PRINT AT 0,1;"▀";AT L,31-A;
"▟";AT A,A-1;"▛";AT L,A;"▙";AT A
,31-A;"▜";AT 11,9;"TIMEX SINCLAIR
"
110 NEXT A
```

mode. Try to follow the pattern. Our thanks to:

Carlos Gonzalez
2601 S.W. 9 St., #2
Miami, FL 33135

sync notes

Paul Grosjean

SYNC at the Keyboard

Mark Fisher leads off our theme section "SYNC at the Keyboard" by revealing the hidden parts of the program line. Then Richard Tracy helps us sort things out, Sharon Aker sets up some arrays, and Edward Beeler shows how to get more into our RAM packs. "Knight's Tour" by Robert Midura lets us watch the computer working out a problem. In "Meteors" Michael Williams shows yet another way of squeezing more into 1K RAM. For TS2068 users, Sharon Aker shares some programming tips, and Robert Hartung develops his method of using the user defined graphics capability. Realizing the power of your computer involves understanding how to use the memory space more fully so V. B. Rice unravels the mysteries of ROM and RAM addressing, and Randall Glidden answers the question of how to use the RAM disabled by your RAM pack. New programming languages also extend the power of your computer as Robert Trelease shows in his discussion of Forth. Finally, we take a break with George Milonas and see how the game Boule can be programmed to the computer.

Coming Issues

Our next theme section will be "SYNC on the Job." We will look at how the computer is being used in work situations —both conventional and unconventional. Other themes under consideration for subsequent issues include: "SYNC at the Arcade," "SYNC Goes Shopping" (a buyer's guide to TS2068 products), and "SYNC in the Classroom."

Timex Developments

Timex Command Cartridge

TS1000 and TS1500 users will now be able to enjoy cartridge software. The Timex Sinclair 1510 Cartridge Player (T-Dock for short) plugs firmly into the expansion port of your computer and allows the use of Timex's new line of cartridge software. The unit measures 3 1/4 x 3 5/8 x 1 1/4 inches, not including the cartridge slot extension. (See photo.)

Timex Command Cartridge Player and Cartridges.

On the TS1500 you need only the T-Dock, but on the TS1000 you must use your 16K RAM pack. We had no problem with RAM pack wobble, but precaution must still be taken. The unit also has a reset button. Some users may want to keep the T-Dock plugged into the computer all the time in order to have the reset capability even though cartridge use is not planned.

The Command Cartridge measures 2 1/2 x 2 3/4 x 3/16 inches. It has a handle on the top for getting a good grip for removal. A board extends out the bottom about 3/16 of an inch. This plugs into the vertical socket on the T-Dock. A sleeve protects the board until the cartridge is inserted in the socket. Although the cartridge looks identical to those for the TS2068, there is no danger of making a mistake because the boards are different and cannot be plugged into the wrong machine.

On the TS1500 you turn the computer power on, the cartridge loads almost instantly, and the program screen comes up. On the TS1000 you must load the program with a USR call. This may take several seconds, but it is significantly faster than tape loading.

The limitation of the T-Dock system is, of course, that you cannot save a program with your data back to the cartridge. Nevertheless at $19.95 the T-Dock promises to save substantial time and relieve much user frustration in loading software.

Cartridge Software

As samples of what is to come from Timex's cartridge software, Timex's winter catalog includes seven cartridges for the TS2068 and four for the TS1000/TS1500 in its list of over 100 software packages. The prices are generally in the $19.95 range. We have seen "Crazybugs" for the TS2068 (see review on p. 32) and for the TS1000/TS1500 "Supermath," "States and Capitals," and "Flight Simulator" (reviewed in the cassette version in *SYNC* 3:2).

Third Party Support

However, more important is that third party vendors will be able to offer programs in the Command Cartridge format. Interested vendors should request information on their official letterhead from Billy Skryme at Timex. Third party vendors may also request (on letterhead) information on the hardware/software guide from Billy Skryme and on schematics from Doug Smith (Timex Computer Corp., Waterbury, CT 06725).

COMPUTER ADD-ONS

Raised stand for TS1000/1500

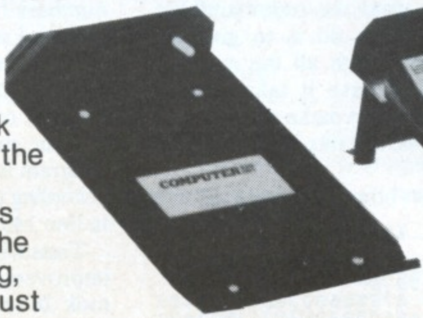
$19.90 Order ref: PBS1

COMPUTER STANDS
Power base

Raised stand for TS1000 and TS1500. Fitted LOAD/SAVE switch, and power ON/OFF for screen clearing. Saves wear and tear on connections. Front lip holds computer in place. No soldering, comes complete with all necessary plugs and cables.
NEW Spectrum Sound, with build in variable sound amplifier. (PBS2).

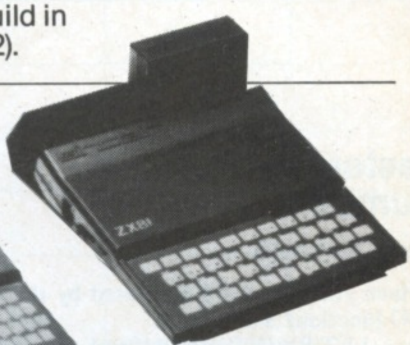
WOBBLE STOPPERS

At last, a fool proof method of avoiding TS1000 whiteouts caused by Ram-Pack wobble. Simply turn the plastic thumbscrew until the Ram-Pack is held firmly against the TS1000. No soldering, no stick-on pieces, just common sense! Flat or tilted models.



Small flat for TS1000/1500 & Ram Pack.
$7.90 Order ref: SW

Small tilted for TS1000/1500 & Ram Pack.
$9.90
Order ref: STW



Large flat or tilted for TS1000/1500 Ram Pack and Printer.
$8.90 flat ref. LW
$10.90 tilted ref: LTW

$76 ME1 64 USA
$25.50 ME1 16 USA

FABULOUS SINCLAIR/TIMEX OFFER FOR TS1000 OWNERS

These are the real thing. TS1000 memory extensions made for the USA and specifically screened to stop radio interference. **Fully Guaranteed for 12 months**

- **no soldering • easy to fit • simply plug in**
- **no loss of memory through wobble or whiteout**

COMPUTER ADD ONS
414 South Evergreen, Arlington Heights, Illinois 60005.

Please send .... ref: ....
enclose check for $ .... payable to Computer Add Ons
Signature .... Date ....
Name ....
Address ....
....
Occupation S2

Add $4.00 Shipping & Handling to all prices.

CIRCLE 4 ON READER SERVICE CARD

just for fun

"Just for Fun" shares short programs that illustrate a point, demonstrate a technique, or show something the author has found interesting. If you have some programs that you want to share, send them to: Just for Fun, SYNC, 39 E. Hanover Ave., Morris Plains, NJ 07950.

Faster Random Numbers

Frank Terranella

Have you ever been frustrated by the RND function? You type in

LET R=INT (RND*10)+1

to generate random numbers in the range of 1 to 10. While an output of 3, 2, 2, 4, 7, 7, 5, 3 is random, it is not pleasing to have the same number crop up so often. What can we do?

The conventional answer involves taking every random number generated and comparing it to all those generated before it and discarding it if it is a repeat. Although this method works, the computer must do a lot of work. This takes a long time. For example, I used the program in Listing 1

Listing 1.

```
  5 REM CONVENTIONAL METHOD
 10 DIM Y(52)
 20 FOR X=1 TO 52
 30 LET Y(X)=INT (RND*52)+1
 40 IF X=1 THEN GOTO 80
 50 FOR Y=X-1 TO 1 STEP -1
 60 IF Y(X)=Y(Y) THEN GOTO 30
 70 NEXT Y
 80 PRINT Y(X);"  ";
 90 NEXT X
```

to generate 52 random, nonrepeating numbers. In SLOW mode it took 6 minutes and 30 seconds; in FAST it still took 1 minute and 28 seconds. Of course, this time is variable depending on how lucky the computer's random numbers are at not repeating.

There has to be a better way.

Listing 2 shows my solution. Since we know all the numbers we want to generate and we simply want the order jumbled, the easiest thing to do is to give the computer a string with all the numbers already in it and have it take random slices of the string. To make the mix a bit more thorough, we can mix up the numbers in the string.

Listing 2 shows how to generate random

Listing 2.

```
 100█REM FASTER RANDOM NUMBERS
 110 DIM A$(52,2)
 120 LET X$="07252609392027184019103747500644513522383331151143174813124516082842461405243402492936522141300423033201"
 130 FOR X=1 TO 52
 140 LET R=(INT (RND*(53-X))+1)*2-1
 160 IF LEN X$=2 THEN LET A$(X)=X$
 170 IF LEN X$=2 THEN GOTO 200
 180 LET A$(X)=X$(R TO R+1)
 190 LET X$=X$( TO R-1)+X$(R+2 TO )
 200 IF VAL A$(X,1)=0 THEN LET A$(X)=A$(X,2)
 210 PRINT A$(X);"  ";
 220 NEXT X
```

numbers from 1-52 (useful, e.g., in card games). I started out with the conventional sort in Listing 1 and waited a minute and a half for it to generate the 52 numbers in random order. I then copied that order into the string in line 120. The rest of the program takes random slices of the string according to the random number generated in line 140.

Tests with this method show a dramatic improvement in speed of execution. It took 26 seconds to generate the same 52 random numbers that took more than 6 minutes in SLOW mode by the other method. In FAST, you can have 52 nonrepeating random numbers in 4 seconds. Not bad.

Of course, if you would rather not type in a long string, you can have the computer create one with all the numbers you need. The results are not as pleasing, but it is less work. Listing 3 shows how.

Listing 3.

```
  10 REM AUTOMATIC RANDOM NUMBERS
  20 LET X$=""
  30 PRINT "HOW MANY NUMBERS?"
  40 INPUT X
  50 IF X>99 THEN GOTO 30
  60 DIM A$(X,2)
  70 CLS
  75 FAST
  80 FOR I=1 TO X
  90 LET X$=X$+("0" AND I<10)+STR$ I
 100 NEXT I
 105 SLOW
 110 FOR I=1 TO X
 120 LET R=(INT (RND*(X+1-I))+1)*2-1
 130 IF LEN X$=2 THEN LET A$(X)=X$
 140 IF LEN X$=2 THEN GOTO 170
 150 LET A$(I)=X$(R TO R+1)
 160 LET X$=X$( TO R-1)+X$(R+2 TO )
 170 IF VAL A$(I,1)=0 THEN LET A$(I)=A$(I,2)
 180 PRINT A$(I);"  ";
 190 NEXT I
```

Frank Terranella, 8 Torne Rd., Sloatsburg, NY 10974.

Polar Lines

Christopher D. Marsh

"Polar Lines" demonstrates a technique for drawing lines. It can be applied, e.g., to graphic displays, laser effects, land boundary line plotting, trigonometry programs. The symbols used in the program are all standard conventions in mathematics.

Type the program in, RUN, and ENTER. A pair of coordinate axes are drawn. To use the program follow these steps:

1) Enter the coordinates of the starting point where the line will begin.

Christopher D. Marsh, 911 Summit Dr., Greensburg, PA 15601.

Microtech Marketing

1418 Lansing Street • Aurora, CO 80010

HANDY ORDER FORM
FILL IN AND MAIL TODAY!

Name ______________________

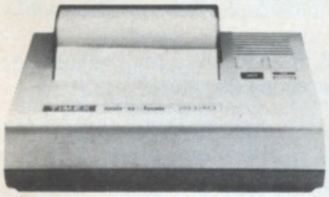
Address ______________________

City ______________ State ________ Zip ________

Phone # (   ) ______________________

2068

MODEL #	QTY.	DESCRIPTION	UNIT PRICE	AMOUNT
OUR CATALOG		Refundable off 1st Order	$ 3.00	
TS 2068		Color Computer 72K	179.95	
TS 2020		Computer Cassette Recorder	49.95	
TS 2050		Modem for 2068	109.00	
TS 2040		Printer	89.00	
COMPUTURF #1		TS 1000, 1500, 2068, 16K Produces 87% Winners at the Horse Races	29.95	
		TOTAL		
		Colorado State residents please add 7.1% sales tax		
		Handling, Shipping and Insurance	5.50	
		GRAND TOTAL		



2040

2050

CHARGE MY VISA MasterCard

Credit Card No. ☐ Visa or ☐ Master Card

Master Card Inter Bank No. (above your name)

Expiration Required Mo. Yr.

METHOD OF PAYMENT: (check one)

☐ PAYMENT ENCLOSED
in the amount of $ ________
(check or Money Order only)

☐ CHARGE CARD
Information filled in at left

Make checks payable to:

MICROTECH MARKETING
1418 Lansing Street, Aurora, CO 80010

SHIP TO: (Fill in only if address is different from above)

NAME ______________________

ADDRESS ______________________

CITY ______________ STATE ________ ZIP ________

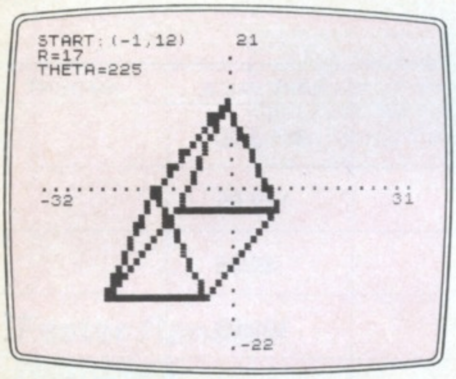
We ship via UPS. Please provide Street Address. UPS will **NOT** deliver to P.O. Box

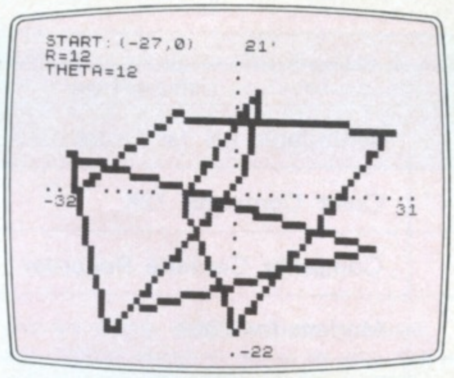
If you are usually out during the day please leave written instructions for UPS men where to leave parcel.

Delivery in 4-6 Weeks

CIRCLE 32 ON READER SERVICE CARD

```
10 REM POLAR LINES
       BY CHRISTOPHER D. MARSH
20 FAST
30 FOR F=0 TO 31
40 PRINT AT 10,F;"."
50 NEXT F
60 PRINT AT 11,0;"-32";AT 11,3
0;"31"
70 FOR F=0 TO 21
80 PRINT AT F,16;"."
90 NEXT F
100 PRINT AT 0,17;"21";AT 21,17
;"-22"
110 SLOW
120 PRINT AT 0,0;"START:";
130 INPUT SX
135 LET SX=INT (SX+.4)
140 PRINT "(";SX;",";
150 INPUT SY
155 LET SY=INT (SY+.4)
160 PRINT SY;")   "
170 PRINT AT 1,0;"R=";
180 INPUT R
190 PRINT R
200 PRINT "THETA=";
210 INPUT THETA
220 PRINT THETA
230 LET Q=THETA*PI/180
240 FOR F=0 TO INT (R+.4)
250 LET X=(F*COS Q)+32+SX
260 LET Y=(F*SIN Q)+22+SY
270 PLOT X,Y
280 NEXT F
290 LET SX=X-32
300 LET SY=Y-22
310 PRINT AT 0,7;INT (SX+.4);",
";INT (SY+.4);")   "
320 PRINT AT 1,0;"            ";A
T 2,0;"          "
330 GOTO 170
```





2) Enter the coordinates, first the X coordinate and then the Y coordinate.

3) Enter the length of the line (R).

4) Enter the angle (THETA) along which the line will be plotted from the starting point. Where each line ends, the next begins. The computer displays each new starting point.

Input Anywhere II

Brian Kautz

Another way of inputting anywhere (Matt Dralle, "Input Anywhere" *SYNC* 3:6) is shown in the program below. It demonstrates a very nice way of inputting information into other programs.

This technique can handle alphabetic or numeric data and can accept up to 31 characters. Input is accomplished just like the normal input by entering the alphanumeric data and then pressing ENTER. A delete function can be added without much difficulty.

Setting the values of I and J moves the prompt around. Beware! The larger the value of I, the smaller the input can be. In line 15 any character that you want can be used as prompt.

```
10 REM REAL PROMPT
20 LET W$=""
30 LET X=10
40 LET Y=13
50 PRINT AT 9,0;"ENTER A 1-19
DIGIT";AT 10,0;"LONG NUMBER. ";
60 PRINT "■"
70 LET Z$=INKEY$
80 IF Z$="" THEN GOTO 70
90 IF Z$=CHR$ 118 THEN GOTO 14
0
100 PRINT AT X,Y;Z$;"■";
110 LET Y=Y+1
120 LET W$=W$+Z$
130 GOTO 70
140 PRINT AT X+(Y=32),Y*(Y<>32)
;" "
150 PRINT AT 12,5;"THANK YOU"
160 PRINT AT 14,5;"YOU ARE ";W$
;" YRS OLD."
170 LET W=VAL W$
```

Brian Kautz, 35 Central Blvd., Camp Hill, PA 17011.

Screen Strings

Robert J. Midura

"Screen Strings" illustrates the technique of storing the screen display in a string variable. This has the advantage of being able to print almost instantly with one simple PRINT statement. No time consuming FOR-NEXT loops are needed.

To use the program, put the computer in SLOW mode, type RUN 100 and ENTER. When the program stops with the message 0/160, type PRINT P$ to see the stored string.

```
10 DIM P$(704)
20 LET X=1
30 LET L=PEEK 16396+256*PEEK 1
6397+1
40 FOR L=L TO L+725
50 IF PEEK L=118 THEN GOTO 80
60 LET P$(X)=CHR$ PEEK L
70 LET X=X+1
80 NEXT L
90 RETURN
100 FOR I=1 TO 22
110 PRINT TAB INT (RND*20+1);"T
EST PATTERN"
120 NEXT I
130 FAST
140 GOSUB 10
150 CLS
160 SLOW
```

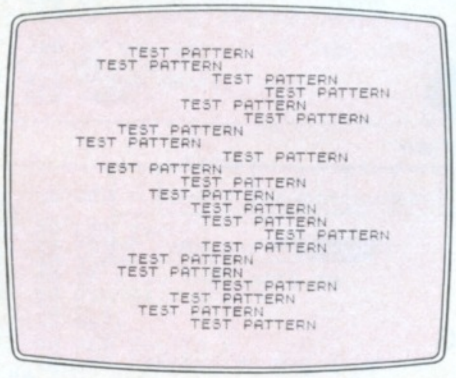
Line Notes

10-90: Store a screen image in the string P$. The length of P$ is the number of character positions on the screen: 22 lines by 32 columns equals 704.

50: Excludes the ENTER characters. TS2068 users must substitute 13 for 118.

60: The variable L addresses the display file and stores each character in P$.

100-160: Demo program.



Robert J. Midura, 19 Merrifield St., Worcester, MA 01605.

Random Squares

J. C. Newton

"Random Squares" fills your screen with randomly placed overlapping squares. The results suggest some modern types of art.

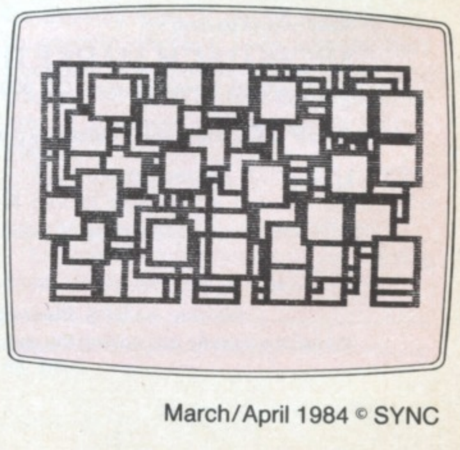
J. C. Newton, North West River, Labrador, Canada A0P 1M0.

Line 30 lets you stop without leaving any incomplete squares by pressing the A key.

With 1K RAM only a limited number of squares can be displayed.

```
3 REM "SQUARES"
5 DIM A$(4,4)
10 LET A$(1)="▛▀▀▜"
12 LET A$(2)="▌  ▐"
14 LET A$(3)="▌  ▐"
16 LET A$(4)="▙▄▄▟"
20 LET S=INT (RND*29)
22 LET T=INT (RND*17)
25 FOR H=0 TO 3
27 PRINT AT T+H,S;A$(H+1)
28 NEXT H
30 IF INKEY$="A" THEN STOP
35 GOTO 20
```

Graphics notes:
10: E;7;7;R
12, 14: 5; space (2);8
16: W;6;6;Q



Call our NEW Demo Hot-Line and hear it for yourself!
(212) 625-6220

■ The Zebra-Talker unlimited vocabulary voice synthesizer uses the famous VOTRAX SC01 phoneme synthesizer with 4 programable pitch levels and automatic inflection. It produces all the phonemes required for English speech.

■ Includes assembled and tested Zebra-Talker, 1400-word Votrax dictionary, instruction manual, editing and demo software. Features bus expansion connector for "piggy-back" expandability.

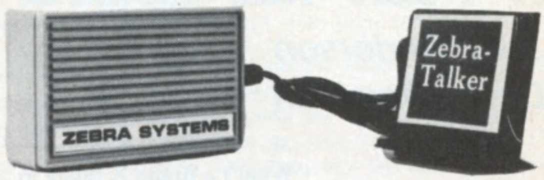
■ Audio output plugs into any audio system, or order our #C145 Amplifier/Speaker module with volume control.

NEW! ZTALKER 1.2 TEXT TO SPEECH SOFTWARE ... Only $19.95
Unlimited vocabulary including: words, letters, symbols and numbers. Talking is as simple as typing "TESTING 1 2 3". Requires 16K RAM.
ORDER C146

$59^{95} VOICE SYNTHESIZER

SPEAKER / AMPLIFIER
ORDER C145 .... $12.50

Zebra-Talker
FOR THE TS1000 & ZX-81
ORDER C141 ...... $59.95



BLIPPO

- TS1000 ZX81 Sound effects generator
- Add amazing sound effects to your games
- Easy to program in BASIC
- Wide frequency range
- Features bus expansion connector
- Assembled & Tested, with Demo software

C122 BLIPPO WITHOUT SPEAKER.... $19.95
C124 Attractive Speaker Module $5.00

GREEN SCREEN

The Green Screen is a specially molded plastic optical filter which relieves eye strain by enhancing character contrast and by eliminating the components of white light which cause eye fatigue. The Green Screen will fit any 12" TV or monitor and can easily be trimmed with a pair of scissors to fit any smaller size.
ORDER C140 .... $12.95

JOYSTICK ADAPTOR

The ZEBRA Joystick Adaptor allows you to connect any Atari compatible joystick to your ZX81 or TS1000. It comes assembled and tested with simple instructions to make joystick versions of most action games. Simply plugs into the back of your computer and features an expansion connector for piggybacking. Use your own Atari compatible joystick or add $6.50 for a genuine Atari Joystick. **FREE PATCHKIT #1 shows patches for TIMEX Flight Simulator and other popular games.**

C120 ZEBRA JOYSTICK ADAPTOR ... $19.95
C121 ATARI JOYSTICK................ $ 6.50
C115 6 JOYSTICK DEMO GAMES CASSETTE .... $ 9.95

ZEBRA LIGHT PEN

- Fun to experiment in BASIC
- Complete package includes:
  - Light Pen
  - Computer Interface Module
  - Demo Program Listings
- Assembled & Tested

ORDER C109 $24.95

ALL ZEBRA PRODUCTS HAVE OUR 15-DAY MONEY BACK GUARANTEE

TIMEX sinclair — WE STOCK ALL TIMEX'S SOFTWARE TITLES AND ACCESSORIES FOR THE TS2068. CALL US FOR PROMPT SERVICE AT COMPETITIVE PRICES.

HARDWARE
TS2040 Printer.............. 87.50
C143 6-Rolls Printer Paper... 11.90
TS2050 Modem............ 111.50
TS2020 Cassette Recorder .. 49.95
TS2068 72K Color Computer182.45
TS2090 Command Stick..... 14.95
TS1500 16K Computer 79.95
TS1510 Cartridge Adapter ... 19.95
TS1000 2K Computer 49.95
TS1016 16K RAM Pack 49.95
All titles for TS2068 only →

ENTERTAINMENT CASSETTES
64001 Androids............. 19.95
64002 Penetrator 19.95
64004 Casino I.............. 15.95
64005 Crossfire............. 19.95
64012 Fun Golf 15.95
64013 Chess 19.95
64014 Hungry Horace 19.95
64015 Horace Goes Skiing... 19.95
64019 Horace & The Spiders . 19.95
64021 Blind Alley 19.95
64023 Crazybugs 19.95

TIMEX HOME CASSETTES
62000 Budgeter............. 24.95
62001 Home Improvement Planner 19.95
62003 Auto Analyzer 24.95
62004 Checkbook Manager .. 19.95

TIMEX BUSINESS CASSETTES
61000 Vu-Calc 24.95
61001 Vu-3D................ 24.95
61002 Vu-3D................ 24.95
61003 Quadra Chart......... 19.95

TIMEX EDUCATION CASSETTES
63000 Flight Simulator 24.95
63001 States and Capitals ... 15.95
63016 Math Wizardry I....... 15.95
63017 Math Wizardry II 15.95
63019 Wordcross 15.95
63020 Word Play I........... 15.95
63021 Word Play II 15.95
63025 Language Usage...... 19.95
63028 Capitalization Master . 15.95

EZ-KEY CASSETTE
EZ001 Upload 2000 .... 19.95

TIMEX COMMAND CARTRIDGES
72000 Budgeter............. 34.95
73000 Flight Simulator 39.95
73001 States & Capitals 29.95
74004 Casino I.............. 29.95
74005 Crazybugs 29.95

SOFTSYNC TS-2068 CASSETTES
SS001 Voice Chess 24.95
SS002 Cyberzone........... 19.95
SS003 Gulpman 19.95
SS004 Personal Accountant . 21.95
SS005 Zeus Assembler...... 19.95

KEYBOARD BEEPER

FOR THE TS1000/TS1500

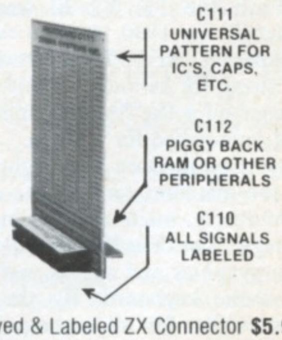
Experience for yourself how much the Keyboard Beeper improves the use of your TIMEX keyboard. Simply plug it into your computer's expansion connector and hear a beep whenever a character is accepted by the computer's keyboard input routine. No software is required. Features expansion connector for RAM etc.

BONUS 1 - You can trigger the beep from Basic. Great for program prompts, timers,
ORDER #C133 .... $19.95

THE BEST BOOKS

C134 ELECTRONICALLY SPEAKING: COMPUTER SPEECH GENERATION by John Cater. 230 pages. Very enjoyable reading. Useful & up-to-date **$14.95**
C135 LEARNING TIMEX SINCLAIR BASIC by David Lein. 350 pages. Great for beginners **$14.95**
C103 NOT JUST 30 PROGRAMS FOR THE ZX81. Interesting programs with explanations . **$ 9.95**
C102 BYTING DEEPER INTO YOUR TS1000. Excellent presentation with 37 tutorial programs **$12.95**
C105 UNDERSTANDING YOUR ZX81 ROM by Ian Logan. Good study of ZX81 & assembly language..................... **$14.95**
C108 THE COMPLETE TS1000/ZX81 ROM DISASSEMBLY by Ian Logan. A must for advanced users **$19.95**
C106 MACHINE LANGUAGE MADE SIMPLE FOR YOUR SINCLAIR & TIMES TS1000. **$14.95**
C107 THE INS & OUTS OF THE TS1000 & ZX81. Excellent hardware manual **$12.95**

EXPERIMENTING?



C110 Keyed & Labeled ZX Connector **$5.95**
C111 Universal Protctyping Board .. **$9.95**
C112 Expansion Connector **$2.50**

ZEBRA SYSTEMS, INC.

78-06 Jamaica Avenue, Woodhaven, New York 11421 (212) 296-2385

ADD $3.00 PER ORDER FOR SHIPPING.
WE ACCEPT VISA, MASTERCARD, CHECKS, M.O.
C.O.D. ADD $3.00 EXTRA.
N.Y. RESIDENTS ADD SALES TAX.
OVERSEAS, FPO, APO, ADD 10%.
DEALER DISCOUNTS AVAILABLE.

CIRCLE 69 ON READER SERVICE CARD

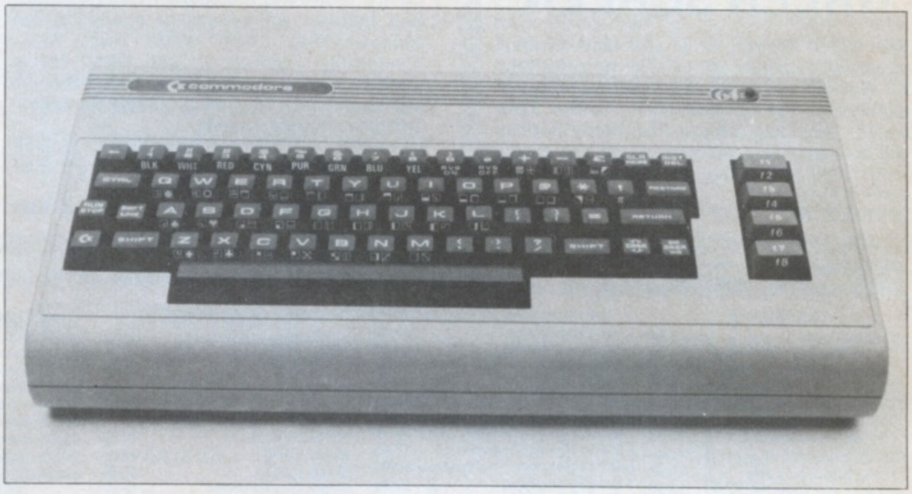
in and out of sync

John Anderson
David Grosjean

What's a Brand X doing in *SYNC* Magazine?
With improving technology and intensifying competition in the small computer market, more and more computers are available at prices within a few steps of the Timex/Sinclair units. Our sister publication, *Creative Computing,* evaluates many of these systems. We would like to share these reviews with those of you considering another computer.
In addition, we will sometimes take a program or two and show what it would be like to write and run the program on the Brand X computer compared to the Timex/Sinclair. You will probably find these tutorials a useful aid for converting programs from other sources to your Timex/Sinclair computer.

The Commodore 64

John Anderson



The Commodore 64 is the best-selling machine in its class, and for very good reason. At under $200, it is quite a buy. With 64K RAM, superlative color graphics, and sophisticated sound synthesis capability, the C-64 packs a lot of punch. It also has a ROM cartridge slot, full-stroke keyboard with four function keys, and an expansion bus.

Commodore, which began years back as a business furniture and calculator company, was among the first manufacturers to offer an assembled microcomputer. It introduced the Commodore Pet in 1978. And despite the fact that the initial machines were slightly flaky and sported the most unfriendly keyboards around, the Pet remains a sight in many homes and schools. It is at the least remembered fondly by many a micro enthusist.

When the 5K Vic-20 was first introduced in 1980, it met with little enthusiasm. At $300, the machine was too cheap for serious hobbyists, and too expensive for the fledgling mass market. So it languished for a while.

But Commodore kept right on pushing the machine, and, to the chagrin of the industry, made cost-cutting its basic strategy. The Vic-20 offered a taste of color graphics and sound, and sported a full-stroke keyboard. By the time the mass market for microcomputers really began to heat up (Christmas of 1981), the Vic-20 was selling for $200. It had some decent software, too, though mostly from third-party manufacturers. And so it began to catch on.

By Christmas of 1982, over a million Vic-20s had been sold, and they continue to be sold today, at prices well under $100. Without a doubt, the Vic was the first true mass market machine. It set Commodore afloat.

Meanwhile, engineers at Commodore continued to improve the Vic, giving it advanced capabilities and a quantum leap in RAM. In early 1983, the Commodore 64 was christened. It was news even at a list price of $595, but it did not stay there for long. Soon after the introduction of the C-64, the first volleys of a bloody, and now legendary, price war were heard.

During this time, the price of the Timex Sinclair dropped from $150 to $50. Commodore prices dropped just as dramatically. And the real winner of the hardware wars was you, the consumer.

One of the strong points of the C-64 is its processor, the 6510. Its architecture is modeled after the 6502, which is the heart of the Apple and Atari micros. As a result, high quality translations of Apple and Atari games and application software have become available. This library will continue to grow.

But the Commodore 64 is not without its problems. Never known as a company with a remarkable support system, they have run aground on a few reefs of their own making.

Problems with reliability have plagued the C-64 since its inception, and though these problems seem largely to have been solved, there is still a ghost in the machine. Early 64s suffered from serious video problems. Later runs turned up units that were D.O.A. (dead on arrival).

Then there is Commodore 64 Basic. It was pulled over from the Pet and Vic machines, without much alteration. As a result, many of the advanced capabilities of the C-64 cannot be accessed from Basic without recourse to complicated PEEKs and POKEs. *Simon's Basic,* an improved language with powerful commands that allow Basic to finally latch in to the potential of the C-64, has just been released. It is the brainchild of a sixteen-year-old programmer from Britain. Not too many serious C-64 enthusiasts will do without it for long. Programming graphics and sound is just too tedious from plain old Basic.

Still, the C-64 can do many things that the more expensive machines can do, and has grown to be a market presence that cannot be ignored. And if the Coleco Adam ever manages to become a marketing force itself, Commodore is ready to cut its prices once again. It's hard to argue with a strategy like that. At $99.95, the C-64 would be hard indeed to beat.

Changing Colors

David Grosjean

Although many computers have the ability to use colors, the system for using the colors varies from machine to machine and sometimes becomes quite confusing. On the TS2068 and the JR200, for example, the system is rather simple. In this issue we compare color changing on the TS2068 and the Commodore 64 and on the TS1000 which is limited to black, white, and gray.

Cursor Color

Changing the color in which the cursor prints is easy on both the TS2068 and the C-64. On the C-64, color is produced by holding down the CTRL (control) key and pressing one of the number keys 1 through 8, e.g., CTRL 3 produces red. Using the Commodore key instead of the CTRL key produces 8 more colors.

If you are typing in direct mode, changing the color immediately starts the cursor printing that color; however, if you are typing a PRINT line of a program and you wish to change the color, the computer inserts a control character instead of directly changing color. E.g., type

30 PRINT "(CTRL 2) SYNC"

Where you hit CTRL 2, there is an inverse E. When you run this, that inverse E does not show up, but "SYNC" will be in white.

Producing the same effect on the TS2068 is easier. No matter where you are typing, if you change color, the cursor starts to print in that color; there are no control characters. The colors are on the number keys, just as on the C-64. To change colors, push the CAPS SHIFT key and the SYMBOL SHIFT key to get the E cursor. Hold down the CAPS SHIFT key and press the number key of the color you want. You are ready to type. Try the same test used on the C-64.

Borders and Backgrounds

You can also change the color of the border of the screen and the background on both computers.

On the C-64, POKEing a number (1-15 each corresponding to a different color) into location 53280 changes the color of the border. E.g.,

40 POKE 53280,2

changes the border color to white. Similarly, POKEing into location 53281 changes the background color.

The following program simply changes the colors of the border and background incrementally with a loop (x). Line 60 is a delay loop.

C-64:

```
10 X=X+1
20 IF X>15 THEN X=0
30 POKE 53280,X
40 POKE 53281,X+1
50 GOTO 10
```

The TS2068 has several commands that make changing colors easy. To change the border, simply type BORDER and the number of the color. Also, the background can be changed with the command PAPER and the color number. The background, however, does not change until the screen has been cleared. E.g., type in

10 BORDER 1:PAPER 2:CLS

RUN this. You will see that the border is blue and the background is red. Add these lines:

20 PAPER 7:PRINT "In and out of Sync"

30 PRINT "colors"

RUN this. The background is red, but the background of the letters is white. Clear the screen (CLS and ENTER) and the entire background is white.

Individual Characters

You can control the color of individual characters on the screen. The C-64 has a color memory that stores the color of every character on the screen. To change the color of any one character, just POKE the new color into the correct location in this color memory which starts at location 55296 and is 1000 locations long (the screen has 40 x 25 characters). The following program fills the screen with circles and then changes the color of a randomly chosen circle.

C-64:

```
100 FOR I=1 TO 960
110 PRINT "O";
120 NEXT I
130 POKE 55297+INT(960*RND(1)),
    INT(16*RND(1))
140 GOTO 130
```

Continued on page 55.

The Anatomy of a Program Line

Mark Fisher

You can communicate with the machine in letters and numbers, but it has to do a lot of extra work to accommodate you.

Program listings on the ZX/TS computers are available on command. With either LIST or LLIST, the program in the machine is displayed for you to study or change. But not everything in the program file is displayed! Three elements are censored from the Listings you see.

But ignorance is bliss, you may say. To an extent you would be right. For most programming chores, understanding the structure of the ZX/TS program file is unnecessary. There are times, however, when you may want to work with the program file itself, e.g., when renumbering lines or trying to save space, and, at these times, knowledge of the structure is important.

Computer Structure

This paper is a tool for holding information. The shapes written on it (letters) mean different things as they are combined (words and sentences).

The computer is also a tool for holding information, but it is built in a different way. There are two main differences: 1) it holds binary numbers only, and 2) it is linear, like a string of beads, rather than two dimensional as this paper is. For the convenience of the human users, both of these attributes have been "covered up." You can communicate with the machine using letters and decimal numbers, and receive its replies in the same way, but the machine is doing a *lot* of extra work to accommodate you. When we look at the structure of the computer, though, we must keep those differences in mind.

Mechanics

The computer stores numbers in a long string. Each number, in reality, is a collection of 32 transistors, formed into eight switches, or *bits.* These eight bits together are called a *byte.* As it happens, eight switches can be arranged in 256 different ways, from 00000000, 00000001, 00000010, . . . to 1111111. We can think of each arrangement as a number, between 0 and 255. Each of these eight bit numbers has a fixed position in the string of numbers; these positions are called addresses. Not all addresses have switches presently attached—it is a little like a new Levittown, with empty streets stretching off into the distance. In the basic TS1000, only the addresses from 0 to 8192, and from 16384 to 18432 are used, while the machine could work with addresses up to 65536!

The Program

Type in the program in Listing 1. It will be stored between 16509 and 16607. You may want to change LPRINT to PRINT after you have RUN it once.

Listing 1.

```
  10 REM SAMPLE PROGRAM, NO. 1
  20 FOR N=16509 TO 16608
  30 SCROLL
  40 LPRINT N;" ";PEEK N;TAB 10;
CHR$ PEEK N
  50 NEXT N
```

Otherwise, type it in *exactly* as written —the success of your future POKEs depends on it.

This program uses two features of Basic to help us examine the machine's memory: PEEK and CHR$. PEEK, followed by an address, will give us the decimal value of the number at that address. The machine can interpret that number in several ways—(as text, as a binary number, or as machine code). We can use CHR$ to command the machine to translate the number into text. POKE, the companion command to PEEK, lets us change the value stored in RAM.

Tokens

RUN the program and look carefully at the results. "Wait!" you say, "each address is supposed to hold one number, and some seem to hold whole words." You are seeing *tokens.* When the machine translates its memory into text, some numbers are assigned whole words, rather than individual characters. A list of all the tokens is in the character list in your User's Manual.

The Powerful 118

The next thing you notice is that the nice, neat program lines are gone; the numbers run along without an apparent break. However, two kinds of breaks are there. Note how each line of the program has a 118 to conclude it (each line of the screen display has one also, but that is a subject for another time). When the LIST routine is drawing the program to the screen, it uses the 118 markers to signal the end of a line.

We can have some fun with that. Type in POKE 16520,118 and ENTER. The LIST routine now thinks that the REM is in two parts! The program will still run; the machine uses a different method of keeping things straight as it RUNs.

Now try POKE 16521,118 and ENTER. List the program. Where did it go? It is still there; try RUN to prove it. The LIST routine interprets two 118's to-

Mark Fisher, 8014 Flower Ave., Apt. #B, Takoma Park, MD 20912.

gether as the end of the program file. If you want to see the rest of the program again, POKE 16520,0 and 16521,0

The Silent 126

The machine has another flag hidden in the listing. RUN the program again, and watch the LISTing of line 28 (addresses 16536 to 16566). The number 16509 appears, starting at 16543. You see the character codes for 1, 6, 5, 0, and 9, but there are six bytes of garbage before you get to the "TO." The 126 tells the computer that a five byte binary number follows. This number is inserted after each decimal number you include in your program line, as the line is copied from the bottom of the screen into the program area. When you press EDIT, these numbers are stripped out.

Try POKE 16520,126. LIST now thinks that the letters "PROGR" are a binary number and hides them from you. They are still there; RUN the program again to prove it. Now LIST it again and pull line 10 down with EDIT. It looks the same as in the listing, but "PROGR" is now deleted. Press ENTER to replace it, and RUN to prove it.

This will happen to machine code if it happens to contain a 126 (which, unfortunately, is a very important command) and is pulled down with EDIT.

Try POKE 16548,0 and LIST. This will remove the 126 from behind the decimal number 16509, and LIST will provide the CHR$ for the five following bytes, rather than skipping them as it normally does. Sometimes, a bad load will change 126 to something else, and the program will not operate correctly. The symptom, of course, is six characters of garbage following a good number in the LISTing. If you like, you can add a program such in Listing 1. Identify the address of the missing 126, and replace it with a POKE.

More Anatomy Line Numbers . . .

There are two more "structures" that help the machine navigate through the program file. The first is the line number. This is stored in an almost logical way. Remember that the largest number that an address can hold is 255. How would it hold a line number larger than that? Let's see.

EDIT line 50, change its line number to 300, and RUN. The line number for

Figure 1.

```
16509 0             16534 29  1       16559 36  8         16584 211 PEEK
16510 10  ▒         16535 118 ?       16560 126 ?         16585 51  N
16511 23  *         16536 0           16561 143 ▛         16586 25  ;
16512 0             16537 20  =       16562 1             16587 194 TAB
16513 234 REM       16538 27  .       16563 192 "         16588 29  1
16514 56  S         16539 0           16564 0             16589 28  0
16515 38  A         16540 235 FOR     16565 0             16590 126 ?
16516 50  M         16541 51  N       16566 118 ?         16591 132 ▜
16517 53  P         16542 20  =       16567 0             16592 32  4
16518 49  L         16543 29  1       16568 30  2         16593 0
16519 42  E         16544 34  6       16569 2   ▪         16594 0
16520 0             16545 33  5       16570 0             16595 0
16521 53  P         16546 28  0       16571 231 SCROLL    16596 25  ;
16522 55  R         16547 37  9       16572 118 ?         16597 214 CHR$
16523 52  O         16548 126 ?       16573 0             16598 211 PEEK
16524 44  G         16549 143 ■       16574 40  C         16599 51  N
16525 55  R         16550 0           16575 24  /         16600 118 ?
16526 38  A         16551 250 IF      16576 0             16601 0
16527 50  M         16552 0           16577 225 LPRINT    16602 50  M
16528 26  ,         16553 0           16578 51  N         16603 3   ▄
16529 0             16554 223 TO      16579 25  ;         16604 0
16530 51  N         16555 29  1       16580 11  "         16605 243 NEXT
16531 52  O         16556 34  6       16581 0             16606 51  N
16532 27  .         16557 34  6       16582 11  "         16607 118 ?
16533 0             16558 28  0       16583 25  ;         16608 118 ?
```

On the TS2068 the program file no longer starts at 16509 or any fixed address, but PEEKing 23635 and 23636 will give the starting address.

that line is held at addresses 16601 and 16602. The computer has taken the line number 300, seen how many 256's would go in it (one), and put the remainder in the next address. Try POKEing 16601,10 and LIST. The first number now represents 2560, and the remainder from before is added in, making 2604.

And Line Length

There are two more bytes to account for. These are extremely important for the machine, as they tell the computer how long the present line is, and thus, how many addresses to move to get to the next line,

RUN, and look at line 10. Address 16511 holds 23. Now starting at the first byte of the text of the line (16513), count down 23. You find the 118 at the end! Many computers only allow lines of less than 256 characters (can you guess why?), but the ZX/TS machines support lines of essentially any length. If the length is greater than 255, the same operation as the line number is performed, with one byte getting the number of 256's in the length, and the other getting the remainder. There is one catch though: the two bytes are in reverse order, to speed the handling of the numbers by the Z80.

If You Have a 2068

The 2068 is organized in a similar way to the 1000 and the 1500, however, it was designed to provide a wider variety of abilities. Therefore its overall organization is much more complex. The organization of the program file, though, is very similar. The changes are:

1) The program file no longer starts at 16509, or any fixed address for that matter. Finding the program is simple, however, as the address where it starts is stored at a fixed address. The starting address of the program can be found by using

PEEK 23635+256*PEEK 23636

2) The character codes have been changed to conform to the codes used by other computers (ASCII code). This is not too critical—CHR$ will still give you the meaning of the given addresses—but the 1000's end line and number markers (118 and 126) had to be changed. The new end line marker is 13, and the new number flag is 14.

3) ASCII codes below 32 are not printable. Like the end line (also called line feed or newline) or number marker, the other numbers below 32 generally give the machine instructions and will affect only how you see the characters that follow if you attempt to print them.

We can still look into the anatomy of the 2068's program lines, though. The program in Listing 2 will do it.

Run it, and now go back to the TOKENS section, and follow the exploration of your machine. You will have to substitute 13 for 118, and 14 for 126 as you go along. Try hitting inverse video somewhere in the listing, and look for where it appears in the program printout.

Now you have seen part of the machine's structure that was hidden before. Like a sleek car, it sometimes pays to lift the hood and appreciate the effort that went into the works under there as well.

Listing 2.

```
  10 REM 2068 anatomy
  20 FOR x=PEEK 23635+256*PEEK 2
3636 TO PEEK 23627+256*PEEK 2362
8
  30 PRINT x;" ";PEEK x,: IF PEE
K x>31 THEN PRINT CHR$ PEEK x;
  40 PRINT
  50 NEXT x
```

Figure 2.

```
26710 0
26711 10
26712 14
26713 0
26714 234      REM
26715 50       2
26716 48       0
26717 54       6
26718 56       8
26719 32
26720 97       a
26721 110      n
26722 97       a
26723 116      t
26724 111      o
26725 109      m
26726 121      y
26727 13
26728 0
26729 20
26730 75       K
26731 0
26732 235      FOR
26733 120      x
26734 61       =
26735 190      PEEK
26736 50       2
26737 51       3
26738 54       6
26739 51       3
26740 53       5
26741 14
26742 0
26743 0
26744 83       S
26745 92       \
26746 0
26747 43       +
26748 50       2
26749 53       5
26750 54       6
26751 14
26752 0
26753 0
26754 0
26755 1
26756 0
26757 42       *
26758 190      PEEK
26759 50       2
26760 51       3
26761 54       6
26762 51       3
26763 54       6
26764 14
26765 0
26766 0
26767 84       T
26768 92       \
26769 0
26770 204      TO
26771 190      PEEK
26772 50       2
26773 51       3
26774 54       6
26775 50       2
26776 55       7
26777 14
26778 0
26779 0
26780 75       K
26781 92       \
26782 0
26783 43       +
26784 50       2
26785 53       5
26786 54       6
26787 14
26788 0
26789 0
26790 0
26791 1
26792 0
26793 42       *
26794 190      PEEK
26795 50       2
26796 51       3
26797 54       6
26798 50       2
26799 56       8
26800 14
26801 0
26802 0
26803 76       L
26804 92       \
26805 0
26806 13
26807 0
26808 30
26809 30
26810 0
26811 245      PRINT
26812 120      x
26813 59       ;
26814 34       "
26815 32
26816 34       "
26817 59       ;
26818 190      PEEK
26819 120      x
26820 44       ,
26821 58       :
26822 250      IF
26823 190      PEEK
26824 120      x
26825 62       >
26826 51       3
26827 49       1
26828 14
26829 0
26830 0
26831 31
26832 0
26833 0
26834 203      THEN
26835 245      PRINT
26836 194      CHR$
26837 190      PEEK
26838 120      x
26839 59       ;
26840 13
26841 0
26842 40       (
26843 2
26844 0
26845 245      PRINT
26846 13
26847 0
26848 50       2
26849 3
26850 0
26851 243      NEXT
26852 120      x
26853 13
26854 248      SAVE
```

An Assortment of Sorts

William F. Tracy

When you must order and retrieve information, you can recall data in minimal time if the data has an orderly placement. In a program to store names and telephone numbers consider how you would locate someone if the data are in random order, e.g., Jones, Smith, Allen, Zimmerman, Travis. Location of a specific name could be made by starting at the first of the list and examining each entry until the correct one is located. Such a process is known as a linear search (see *SYNC,* 3:6, for a machine language linear search). Linear searches are easy to conduct, but time consuming to execute when the material being searched for is longer than a few items. The usual way for humans to order and search lists is either alphabetically for names or by order of magnitude for numbers.

Comparing Sorting Types

Any system of file handling will usually include routines for sorting (arranging the data in some sequential order) and searching. In this paper, four sorts will be examined: Bubble, Float, Shell, and Quick. There are others but these were chosen because they are popular and because they represent slow, intermediate, and fast, as well as simple and complex algorithms. Let's look at how they work and the speed at which they order data.

The routines were run with numeric and alphanumeric data and timed with a stop watch. Three runs were then averaged. The first trials involved sorting numbers under two conditions: 1) a "worst case" situation in which the original order was from high to low and the order to be sorted into was from low to high; and 2) a random order grouping. String timing consisted of sorting strings

William F. Tracy, 200 Retreat Court, W., Hermitage, TN 37076.

Listing 1 gives the program for sorting numbers, and Listing 2 for words. To use the programs, type in the listing you want. Note that the line numbers are parallel so that the other listing can be entered with minimal editing.

After you have typed in the listing, press RUN and ENTER. The program will then ask how many items are to be sorted. If words are to be sorted, the program asks for word length. The program will then generate random data.

Listing 1.

```
  10 REM "SORTS"
  20 FAST
  30 PRINT "INPUT TOTAL NUMBERS
TO SORT"
  40 INPUT NUM
  50 DIM U(NUM)
  60 CLS
  70 GOSUB 1060
  80 LET A=NUM
  90 GOSUB 1100
 100 INPUT W$
 110 REM TIME FROM THIS POINT
 120 IF W$="1" THEN GOTO 180
 130 IF W$="2" THEN GOTO 310
 140 IF W$="3" THEN GOTO 470
 150 IF W$="4" THEN GOTO 670
 160 CLS
 170 GOTO 90
 180 REM ** BUBBLE SORT **
 190 FOR Q=1 TO NUM-1
 200 FOR R=1 TO NUM-Q
 210 LET H=U(R)
 220 LET I=U(R+1)
 230 IF H<=I THEN GOTO 260
 240 LET U(R)=I
 250 LET U(R+1)=H
 260 NEXT R
 270 NEXT Q
 280 GOTO 950
 290 REM ** END OF BUBBLE**
 300 REM
 310 REM *** FLOAT SORT ***
 320 LET Q=U(1)
 330 LET K=1
 340 FOR S=2 TO NUM
 350 IF U(S)<Q THEN GOTO 380
 360 LET Q=U(S)
 370 LET K=S
 380 NEXT S
 390 LET Z=U(NUM)
 400 LET U(NUM)=U(K)
 410 LET U(K)=Z
 420 LET NUM=NUM-1
 430 IF NUM>1 THEN GOTO 320
 440 GOTO 950
 450 REM *** END OF FLOAT ***
 460 REM
 470 REM *** SHELL SORT ***
 480 LET S=1
 490 LET S=S*2
 500 IF S<=NUM THEN GOTO 490
 510 LET S=INT (S/2)
 520 IF S=0 THEN GOTO 660
 530 FOR T=1 TO NUM-S
 540 LET Y=T
 550 LET W=Y+S
 560 IF U(Y)<=U(W) THEN GOTO 620
 570 LET Z=U(Y)
 580 LET U(Y)=U(W)
 590 LET U(W)=Z
 600 LET Y=Y-S
 610 IF Y>0 THEN GOTO 950
 620 NEXT T
 630 GOTO 510
 640 REM *** END OF SHELL ***
 650 REM
 660 REM QUICK SORT ALGORITHM BY
         PERMISSION OF SIMON AND
         SCHUSTER,INC.  FROM:
         THE ESSENTIAL GUIDE TO
         TIMEX/SINCLAIR HOME
         COMPUTERS BY MORSE,
         ADAMSON, ANREP, AND
         HANCOCK.
 670 REM *** QUICK SORT ***
 680 DIM S(NUM,2)
 690 LET P=0
 700 LET L=1
 710 LET R=NUM
 720 LET I=L
 730 LET J=R
 740 LET S=-1
 750 IF U(I)<=U(J) THEN GOTO 800
 760 LET T=U(I)
 770 LET U(I)=U(J)
 780 LET U(J)=T
 790 LET S=-S
 800 IF S=1 THEN LET I=I+1
 810 IF S=-1 THEN LET J=J-1
 820 IF I<J THEN GOTO 750
 830 IF I+1>=R THEN GOTO 870
 840 LET P=P+1
 850 LET S(P,1)=I+1
 860 LET S(P,2)=R
 870 LET R=I-1
 880 IF L<R THEN GOTO 720
 890 IF P=0 THEN GOTO 950
 900 LET L=S(P,1)
 910 LET R=S(P,2)
 920 LET P=P-1
 930 GOTO 720
 940 REM ** END OF SORT **
 950 CLS
 960 PRINT "ARRAY SORTED"
 970 SLOW
 980 FOR X=1 TO A
 990 PRINT U(X)
1000 IF X>19 THEN SCROLL
1010 NEXT X
1020 PAUSE 600
1030 CLS
1040 GOTO 30
1050 REM     GENERATES RANDOM
NUMBERS AND FILLS ARRAY
1060 FOR X=1 TO NUM
1070 LET U(X)=INT (RND*99)
1080 NEXT X
1090 RETURN
1100 PRINT AT 5,15;"PICK SORT";A
T 7,15;"1. BUBBLE";AT 9,15;"2. F
LOAT";AT 11,15;"3. SHELL";AT 13,
15;"4. QUICK"
1110 RETURN
```

21st CENTURY ELECTRONICS

6813 POLK STREET GUTTENBERG, N.J. 07093 (201) 869-2616

COMPATIBILITY for your TIMEX 2068

UNTIL NOW, THE ONLY PROBLEM WITH THE TIMEX-SINCLAIR/2068 MICROCOMPUTER HAS BEEN IT'S INCOMPATIBILITY WITH THE MANY FINE PRINTERS AND OTHER PERIPHERALS OFFERED BY THE COMPUTER INDUSTRY. 21st CENTURY ELECTRONICS NOW BRINGS YOU THE TWO MOST COMMONLY USED DATA INTERFACES IN THE MICROCOMPUTER INDUSTRY. THE CENTRONICS PARALLEL INTERFACE AND THE RS232-C SERIAL INTERFACE CREATED BY AERCO.

CENTRONICS Interface

THE CENTRONICS INTERFACE GIVES COMPATIBILITY WITH ALMOST ANY DOT MATRIX PRINTER AVALIABLE. GRAPHICS SUPPORT IS PROVIDED, AS IS THE ABILITY TO PRINT TEXT IN EITHER THE PRINTER'S NATIVE FONT OR IN THE TS/2068 DISPLAY SCREEN FONT.

69.95

DUAL RS232 Interface

THE RS232-C INTERFACE BRINGS COMPATIBILITY WITH MOST LETTER QUALITY PRINTERS AND MODEMS. TWO SERIAL DEVICES CAN BE CONNECTED AT ONCE, WITH AN ADJUSTIBLE DATA TRANSFER RATE FROM 300 TO 19200 BAUD.

89.95

CABLES INCLUDED

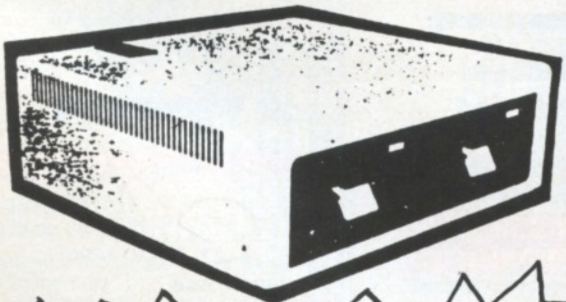
UPLOAD - 2000 by Dave Ornstein

NOW! GET FREE SOFTWARE BY MAKING YOUR TIMEX/1000 PROGRAMS COMPATIBLE WITH YOUR NEW TIMEX/2068. NO HARDWARE INVOLVED. EVERYTHING IS ACCOMPLISHED BY SOPHISTICATED SOFTWARE. YOU WILL BE ABLE TO CONVERT ANY TIMEX/1000 PROGRAM WRITTEN IN BASIC TO RUN ON YOUR TIMEX/2068.

CASSETTE and DOCUMENTATION ONLY **19.95**

FANTASTIC VALUE

DUAL FLOPPY DISC DRIVES



BRAND NEW, dual floppy disc drives made by Digital Equipment Corp. (DEC model no. RX 180AB). This beautiful piece of computer hardware consists of 2 Shugart compatible, 40 track, double density, 5¼" mini floppy disc drives brand new in the case with their own regulated power supply, cooling fan and on/off switch. Each unit also comes with a line cord and full documentation. These were made for DEC, but should be compatible with many other personal computers such as IBM, Radio Shack, TI, and other similiar systems. Naturally, you supply cables and the controller board. The disc drives will run on either 115/230 vac 50/60 Hz. without any modification to the drives. Each unit comes in its original factory carton. All are guaranteed O. K.. A block buster of a buy !

Digital Dual Drive Unit Reg. $500.95

ZX-PROFILE

BY TOM WOODS

NOW NOT ONLY ON CASSETTE BUT ON DISK FOR THE 21st CENTURY DISK DRIVE SYSTEM AND AERCO DISC DRIVE SYSTEMS **ONLY $39.95**

ON CASSETTE $16.95

Interface Board
on board DOS In ROM
Industry Standard IBM Format $189.95

Complete Disk Drive System
Includes Drives, Interface Board
& ZX-PROFILE **$599.95**

All orders shipped postage charge C.O.D. Personal Checks take 3 weeks to clear.
AMX - VISA - MC - Money Order or Certified Check for Turnaround Shipment

CIRCLE 72 ON READER SERVICE CARD

The menu will let you choose the sort type you want. If you are going to time the sort, enter the sort type number and start the timer simultaneously with pressing ENTER. The program will then display the sorted data. You can then modify the programs to accept your own data for sorting.

Listing 2.

```
  10 REM "SORTWD"
  20 FAST
  30 PRINT ,,"INPUT TOTAL WORDS
TO SORT"
  40 INPUT NUM
  50 LET A=NUM
  60 GOSUB 1190
  70 GOSUB 1110
  80 GOSUB 1000
  90 GOSUB 1250
 100 INPUT W$
 110 REM TIME FROM THIS POINT
 120 IF W$="1" THEN GOTO 180
 130 IF W$="2" THEN GOTO 310
 140 IF W$="3" THEN GOTO 470
 150 IF W$="4" THEN GOTO 670
 160 CLS
 170 GOTO 90
 180 REM *** BUBBLE WD SORT  ***
 190 FOR Q=1 TO NUM-1
 200 FOR R=1 TO NUM-Q
 210 LET H$=U$(R)
 220 LET I$=U$(R+1)
 230 IF H$<=I$ THEN GOTO 260
 240 LET U$(R)=I$
 250 LET U$(R+1)=H$
 260 NEXT R
 270 NEXT Q
 280 GOTO 950
 290 REM  END OF BUBBLE
 300 REM
 310 REM *** FLOAT WD SORT  ***
 320 LET Q$=U$(1)
 330 LET K=1
 340 FOR S=2 TO NUM
 350 IF U$(S)<Q$ THEN GOTO 380
 360 LET Q$=U$(S)
 370 LET K=S
 380 NEXT S
 390 LET Z$=U$(NUM)
 400 LET U$(NUM)=U$(K)
 410 LET U$(K)=Z$
 420 LET NUM=NUM-1
 430 IF NUM>1 THEN GOTO 320
 440 GOTO 950
 450 REM  END OF FLOAT
 460 REM
 470 REM *** SHELL WD  SORT ***
 480 LET S=1
 490 LET S=S*2
 500 IF S<=NUM THEN GOTO 490
 510 LET S=INT (S/2)
 520 IF S=0 THEN GOTO 950
 530 FOR T=1 TO NUM-S
 540 LET Y=T
 550 LET W=Y+S
 560 IF U$(Y)<=U$(W) THEN GOTO 6
20
 570 LET Z$=U$(Y)
 580 LET U$(Y)=U$(W)
 590 LET U$(W)=Z$
 600 LET Y=Y-S
 610 IF Y>0 THEN GOTO 550
 620 NEXT T
 630 GOTO 510
 640 REM  END OF SHELL
 650 REM
 660 REM
 670 REM  *** QUICK WD  SORT ***

 680 DIM S(NUM,2)
 690 LET P=0
 700 LET L=1
 710 LET R=NUM
 720 LET I=L
 730 LET J=R
 740 LET S=-1
 750 IF U$(I)<=U$(J) THEN GOTO 8
00
 760 LET T$=U$(I)
 770 LET U$(I)=U$(J)
 780 LET U$(J)=T$
 790 LET S=-S
 800 IF S=1 THEN LET I=I+1
 810 IF S=-1 THEN LET J=J-1
 820 IF I<J THEN GOTO 750
 830 IF I+1>=R THEN GOTO 870
 840 LET P=P+1
 850 LET S(P,1)=I+1
 860 LET S(P,2)=R
 870 LET R=I-1
 880 IF L<R THEN GOTO 720
 890 IF P=0 THEN GOTO 950
 900 LET L=S(P,1)
 910 LET R=S(P,2)
 920 LET P=P-1
 930 GOTO 720
 940 REM  END OF QUICK
 950 REM STOP TIME HERE
 960 CLS
 970 LET SW=0
 980 PRINT "ARRAY SORTED"
 990 PRINT
1000 SLOW
1010 FOR Z=1 TO A
1020 PRINT U$(Z)
1030 IF Z>19 THEN SCROLL
1040 NEXT Z
1050 PAUSE 600
1060 CLS
1070 FAST
1080 IF SW THEN RETURN
1090 GOTO 10
1100 REM  RANDOM WORD GENERATOR
1110 FOR Z=1 TO NUM
1120 FOR X=1 TO WD
1130 LET U$(Z,X)=CHR$ INT (38+RN
D*26)
1140 NEXT X
1150 NEXT Z
1160 CLS
1170 LET SW=1
1180 RETURN
1190 CLS
1200 PRINT ,,"INPUT WORD LENGTH"
1210 INPUT WD
1220 DIM U$(NUM,WD)
1230 FAST
1240 RETURN
1250 PRINT AT 5,7;"SELECT SORT"
1260 PRINT AT 7,7;"1. BUBBLE";AT
 9,7;"2. FLOAT";AT 11,7;"3. SHEL
L";AT 13,7;"4. QUICK"
1270 RETURN
```

Bubble Sort

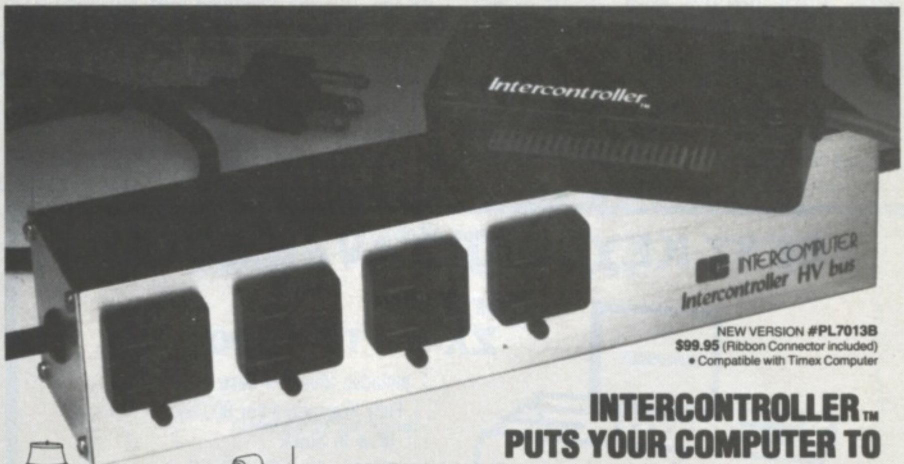
The Bubble Sort is perhaps the most widely known and used sort routine and one of the easiest to follow in its opera-

INTERCOMPUTER INC.

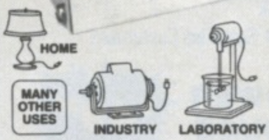
T.M.

Expands the use of your TIMEX SINCLAIR COMPUTERS

All products are compatible with TS1000 and TS1500 Computers



NEW VERSION **#PL7013B**
$99.95 (Ribbon Connector included)
• Compatible with Timex Computer



INTERCONTROLLER™ PUTS YOUR COMPUTER TO WORK FOR YOU AROUND THE CLOCK.

1000's of new applications at home, office, laboratory or shop. INTERCONTROLLER™ enables you to use your computer's power to control any number of tasks, everyday of your life, with just a few lines of BASIC. Four fully programmable outlets. Expandable to 16 with SOFTBOX.™

QSAVE

A Reliable Fast Loading System

- Operates 16 times faster than the Sinclair system (i.e. Loads/Saves 8K in 10 secs; 48K in 110 secs)
- Operates with any memory and on any program length
- Works equally well with AUTORUN programs, BASIC or MACHINE CODE
- This is what you get:
  1. A cassette interface to improve the signal quality. To make loading most reliable an AUTOMATIC LEVEL CONTROL, an OVERLOAD WARNING LIGHT, a SAVE/LOAD SWITCH, an ON/OFF SWITCH are also built into this unit.
  2. Highly sophisticated, superbly easy to use 1/2K software for up to 16K and 64K programs **#PL7001 $37.95**
- **QSAVE** software on **Solid State cartridge** available separately **#PL7001C $24.95**
- Complete unit with software on **Solid State cartridge** **#PL7001B $57.95**

Cartridges need Flexible Ribbon Connector **Ver. 2**

DISASSEMBLER

(Decoding Tool)
On Solid State Cartridge

- Disassemble any assembly language code on memory
- Assembly language games and other programs **#PT4004 $34.95**

Needs Flexible Ribbon Connector **Ver. 2**

PRINTER PACKAGE

- New Memotech DMX Printer
- 80 Column to 136 in condensed form
- Sharp, Precise, high-speed print
- Bidirectional, tractor and friction seed.
- Graphics **#PL7025 $395.00**

MEMOTECH PRODUCTS

MEMOPAK 16K	**#PL7002**	**$35.95**
MEMOPAK 32K	**#PL7003**	**$71.95**
MEMOPAK 64K	**#PL7004**	**$117.95**
HIGH RES.GRAPHICS	**#PL7007**	**$71.95**
RS232 Interface	**#PL7006**	**$71.95**
KEYBOARD	**#PL7011**	**$71.95**
MEMOCALC	**#BS2001C**	**$35.95**
MEMOTEXT	**#PL7014**	**$35.95**
MEMOASSEMBLER	**#PL7015**	**$35.95**

MCODER (16K RAM)

- The first real integer BASIC compiler for the TS1000 computer
- Give your BASIC program the speed of MACHINE language programs
- 75 times and in some cases up to 900 times speed improvement
- Very simple to use:
  1. Load MCODER
  2. Write your BASIC program according to the manual and compile it through the MCODER by using a single PRINT USR command **#PT4001 $18.95**

FLEXIBLE RIBBON CONNECTOR

No More System Crashes

- Connects any RAM expansion to your TS1000
- Prevents usual system crashes caused by the computer or the RAM movement
- 9 inches long flat ribbon cable **#PL7016 $17.95**
- **Version 2 with an EXTRA port for peripherals #PL7016 B $24.95**

ASSEMBLER/DEBUGGER "ZXAD" (16K RAM)

- Write machine code quickly and easily
- Two pass assembler, ZILOG mnemonics, 8 pseudo-OPS
- Debugger includes examine/modifying, breakpoint with register display
- Occupies 7K, comprehensive multi-page manual **#PT4003 $18.95**

CIRCLE 29 ON READER SERVICE CARD

STOPPER

- Enables you to break through any machine code software, list it on the screen, save it on another tape, or print it out
- Very simple to use, load STOPPER into your TS1000 and follow the simple instructions on the screen **#PT4002 $18.95**

DEALERS/DISTRIBUTORS PACKAGE AVAILABLE.
Phone orders: (617) 738-5310 7 days a week.
Call or write to us for FREE brochure and information on our full line of ***software.***

Mail to: Intercomputer, Inc.,
358 Chestnut Hill Ave., Boston, MA 02146
Tel: (617) 738-5310
Telex: 951141COFAR

PART #	QTY.	UNIT	U.S. $	TOTAL
Shipping and Handling for MEMOTECH and QSAVE			4.95	
for EACH PACKAGE			9.95	
Mass. Residents Tax			5%	
			Total	

☐ Visa ☐ M.C. ☐ Check

Account No. ______ Exp. ______

Name ______

Address ______

City ______ State ______ Zip ______

No Shipping & Handling Charges for Cassettes

The Bubble sort is slow; the Float, twice as fast; the Shell, four times faster; the Quick, among the fastest.

tion. It is comparable in time to other sorts when dealing with 15 or fewer items not in a worst case order before sorting. The Bubble Sort, as with other sorts can order data in any direction, from low to high, or high to low. The sorts, described order from smallest to largest in value.

In a sort of N numbers (the variable NUM in the listings) the Bubble Sort works by comparing adjacent numbers. If they are not out of order, it shifts their positions. The first elements n1 and n2 are compared, then n2 and n3, n3 and n4, etc., until the end of the data. When N is reached, the largest value is in the nth position. The sequence is repeated until all comparisons have been made. With a random ordering of the distribution to be sorted, there will be (N*(N**2-1))/2 comparisons. With data already ordered, exchanges = 0. Interestingly, in other sorts, the final pass through the data is often a Bubble Sort.

The Bubble Sort offers the advantage of being simple and easy to understand. As shown it occupies 236 bytes. Its disadvantage is slowness. Table 1 shows a walk through a Bubble Sort for 8 numbers arranged in random order.

Float Sort

In Float Sort, we compare the first element with each succeeding element until a larger value is found. When

Table 1. Bubble sort number flow.

	Order at end of pass						
Data	1	2	3	4	5	6	7
90	28*	01*	01	01	01	01	01
28	01*	28*	28	28	28	28	28
01	47*	47	47	47	32*	32	32
47	90*	88*	56*	32*	47*	47	47
93	88*	56*	32*	56*	56	56	56
88	56*	32	88*	88	88	88	88
56	32*	90*	90	90	90	90	90
32	93*	93	93	93	93	93	93

*Indicates that a position swap occurred.

found, the larger element is then compared to each of the remaining elements. If no larger value is found, that element

Table 2. Float sort number flow.

	Order at end of pass						
Data	1	2	3	4	5	6	7
90	90	56*	56	32*	01*	01	01
28	28	28	28	28	28	28	28
01	01	01	01	01	32*	32	32
47	47	47	47	47	47	47	47
93	32*	32	32	56*	56	56	56
88	88	88	88	88	88	88	88
56	56	90*	90	90	90	90	90
32	93*	93	93	93	93	93	93

*Swap

becomes the last one in the array and the value previously in last position occupies the former position of the larger element. If, after a pass is completed, the first element is the greatest, the first and last elements are swapped.

An advantage of the Float Sort is that, on the average, it is about twice as fast as the Bubble Sort. The number of comparisons is N*(N+1)/2. The routine occupies 281 bytes. Table 1 shows the walk through. With the numbers given, the sort was completed with 28 comparisons, and 4 exchanges in 7 passes.

Shell Sort

Shell Sort involves an interchanging technique in which succeeding passes through the data place the data into a more nearly sorted list. The routine starts by comparing elements at least N/2 positions apart and swapping if the first element is greater than the second. This process is then repeated with successive pairs of elements, the same distance apart, until all elements have been compared. Then the comparison distance is halved for each successive pass through the data.

This routine is, on the average, almost four times faster than the Bubble Sort on 50 item sorts. It occupies 353 bytes. Table 3 shows the Shell Sort example which made 18 comparisons and 4 exchanges in 4 passes.

Quick Sort

Quick Sort works on the premise that it is faster to sort two small arrays rather than one large array. This Sort is reported to be among the fastest of the sorts for disordered data. When data is not disordered but in inverse or nearly sorted order, this routine can be slow. As listed, the algorithm occupies 543 bytes. It is reproduced through the kind permission of Simon & Schuster Publishers and is taken from *The Essential Guide to Timex/Sinclair Home Computers* by Morse, Adamson, Anrep, and Hancock.

The routine operates by dividing the data into two groups about an Index value. The Index is such that the values above it are smaller, and those below are larger. Each group is sorted in turn.

Sorting is accomplished by an upper pointer (I) and a lower pointer (J). The upper pointer starts at U(1) and the lower at U(NUM). If the value indicated by I is greater than J, the values are exchanged and I is advanced one position. If J is greater, I is moved one position down and no exchange occurs. This process continues until the pointers coincide (I=J) and this value, U(I=J), then becomes the Index. One group is set aside and the other group sorted by the above procedure until ordered. The reserved group is retrieved and sorted, and the array is ordered.

With the data shown in Table 4, the quick sort ordered the array with 4 exchanges and 13 comparisons in 6 passes.

Table 3. Shell sort number flow.

	Order at end of pass			
Data	1	2	3	4
90	32*	32	01*	01
28	28	28	28	28
01	01	01	32*	32
47	47	47	47	47
93	93	93	56*	56
88	88	88	88	88
56	56	56	93*	90*
32	90*	90	90	93*

Table 4. Quick sort number flow.

	Order at end of pass					
Data	1	2	3	4	5	6
90-I	32	32	32-I	01		01
28	28	28	28	28-Index		28
01	01	01	01-J	32		32
47	47	47	47	47		47
93	93-I	56	56	56		56
88	88	8-Index				88
56	56-J	93			93-I	90
32-J	90	90			90-J	93
I=1	I=5	I=J	I=1	I=J	I=7	Pointer
J=8	J=7		J=3		J=8	Positions

Table 5. Sort comparison-time/number sorts.

	NUMBERS			STRINGS			
Size:	20		50	20	50	100	400
Order:	Rnd	Inv	Rnd	Rnd	Rnd	Rnd	Rnd
BUBBLE	6	40	34	7	43	176	waiting
FLOAT	4	22	21	4	23	86	1306
SHELL	3	9	11	4	15	62	720
QUICK	3	41	10	3	14	33	162

Time in seconds (rounded off).
Numbers ranged from 1-99.
Strings of length 20.

Conclusions

Which sort is best? The answer depends on the number of items being sorted and their order. From Table 5, it is clear that for sorting large numbers of items, likely to be in random order, the Quick Sort is by far the fastest. If the data is almost sorted or inverse ordered, then the Quick Sort can be slower than the Bubble Sort.

When dealing with 20 items or fewer, time factors are not a major consideration and the programmer considers memory requirements and/or complexity of the algorithm in sort choices. My choice for the "best" all around sort would be the Shell Sort. It is fast, easy to understand, considerate of memory, and relatively free from ordering effects.

Introducing the MTX512

A World Apart from the Ordinary.

It begins with the sense of touch.

With the sleek black metal housing. Cool to the touch. Cool to the advanced circuitry and components contained within.

And the solid feel of people-sized keys set up in a field that gives you room to work and space in which to think.

But the difference only begins with what you see and feel. Where it ends...well, that's really up to you.

In a very practical sense, the only limits you'll experience with the MTX512 are those you choose to accept.

64K To 512K RAM – A Look On The Inside

Take a close look at the MTX512.

We could tell you it offers the greatest performance and versatility of any micro in its price range, but we think you're smart enough to draw your own conclusions.

The design is elegant in its simplicity. Remarkable for the power and complexity it represents. 64K RAM built in, with total expansion to 512K. And that doesn't include 16K of video RAM controlled by its own processor.

Speaking of video, keep in mind this is no ordinary monochromatic, single screen system. The MTX starts off where other micros end up. Delivering vivid screen capabilities with 256 x 192 pixels that crisply define interference-free high resolution graphics. 16 brilliant colors that can be displayed simultaneously. In a format powered by 32 easily movable, user-defined graphics characters. Graphics capabilities you'd find impressive in a system that gives you a single screen to work in. With the MTX, you have eight.

Yes, eight.

Eight definable windows to operate independently or in tandem. And still maintain full screen capabilities. Thus, you can manipulate spread sheets on the MTX and see the impact of changing variables in graphics at the same time. Eight separate windows on the world. We call them Virtual Screens. You'll call them extraordinary.

Far from ordinary as well are the built-in system outputs that come standard on the MTX. The Centronics parallel printer port. The two industry-standard joystick ports. The uncommitted parallel I/O port. The Cassette port with 2400 baud. Separate TV and Video Monitor Ports. The 4-channel sound hifi output. We've even installed a ROM cartridge port for word processing and other dedicated programs.

Interactive Languages And Routines – A Look At The Way All Micros Will One Day Perform

Forget the way all other micros perform. This is the way they should.

Interactively.

With the MTX, you can create and manipulate programs using four different languages in dynamic interaction, all coordinated through the FRONT PANEL DISPLAY. Interweaving elements as you would in creating a symphony.

And take a serious look at the languages housed in the MTX's 24K ROM. MTX BASIC, a more powerful form of BASIC that allows you to use all standard BASIC programs. MTX GRAPHICS, with straight forward commands, eliminates the tedium and difficulty of creating complex graphics programs. NODDY, an 11-command "easy learn" language that can transform real world programming into a child-friendly activity. And MTX ASSEMBLER, which enables sophisticated programming in assembly language. Something else the advanced programmer will appreciate is our ASSEMBLER/DISASSEMBLER, tied to BASIC, which provides unprecedented display and keyboard access to Z80 CPU storage locations, memory and program.

If you're hungry for more, PASCAL and FORTH are also available as add-on ROM packs.

On the keyboard side of things, you'll find a number of operator-oriented features that speed up and ease up the operation of the MTX. The separate numeric pad with quadri-directional cursor control and full editing functions. The eight dual function keys.

The auto repeat function on all alpha-numeric keys. Add to this such programmer-saving features as the use of abbreviated BASIC commands, a built-in syntax verifier, automatic cursor-honing to errors, auto-line numbering and automatic scrolling, and you begin to see the MTX not only opens a lot of doors that other micros leave closed, but speeds you through them as well.

The 160 Megabyte Connection – A Look At The System

To build a good system, quality must be designed in at every level.

We designed the MTX and its complete line of system peripherals using proven, standard components. Striking a strategic balance between power, versatility and dependability. Our Z80A processor, running at 4MHz, gave us the high performance characteristics we were striving for, plus the ability to expand into the MTX Hard Disc, MTX Silicon and Floppy Disc CP/M operated systems. Systems that could provide up to 160 megabyte storage capacity. More power than you'll probably ever need, unless you take full advantage of the MTX's impressive system capabilities.

Systems hookup is as simple as every other MTX procedure. By merely plugging in the twin RS232C Serial interfaces and the Node software, sold optionally, you're ready to create a disc-driven interactive communications network (OXFORD RING®) that can link up to 255 units.

Software? You'll never worry about software availability with the MTX. Dozens of MTX-dedicated programs have already been created, supplementing the vast landscape of CP/M applications software currently available. And advance word of the MTX's technical capabilities has precipitated an MTX software "push" on the part of many leading software manufacturers.

Word Processing For $999 – A Look At A Great Deal

Look first at the capabilities, then at the price.

This is word processing the way it should be. Quick. Easy. Professional. A package that includes the MTX512; the powerful New Word,™ word processing ROM cartridge; and the Memotech DMX80 correspondence quality printer.* An exceptional value!

And that brings us to the bottom line.

A Look At The Price

There's a very simple equation that covers the pricing of the MTX512.

The more engineering you put in a system, the less it will cost to produce. As you've already seen, the MTX is a pure product of advanced, innovative engineering.

Which is why we can sell it for $595.**

And why we can confidently back it up with a full one-year warranty.

Make no mistake. When you turn this page, you'll be returning to a world very different from this one.

A world in which all microcomputers will suddenly seem very different.

Suddenly very ordinary.

For more information about the MTX512, or to find out the location of the MTX dealer nearest you, contact Memotech Corporation, 99 Cabot Street, Needham, MA 02194; or phone (617) 449-6614

MEMOTECH
CORPORATION

CP/M is a trademark of Digital Research, Inc.
New Word™ is a trademark of New Star Software, Inc.
*DMX80 correspondence quality printer suggested retail price $395.
**Suggested retail price.

The Array Advantage

Sharon Zardetto Aker

An array is just another kind of variable, right? Well, yes and no: an array *is* another kind of variable, but it is not *just* another variable.

There are arrays to store numbers, there are arrays to hold strings, and there are lots of things you can do with arrays that you cannot do with simple variables.

Numeric Arrays

A numeric array is like a row of boxes: each is able to hold any number you want to put into it. To tell the computer how many boxes you want to reserve, use a DIM statement, e.g.,

```
DIM B(5)
```

tells the computer to reserve five memory cells. Each cell, or element, in the array is named by the variable (any single letter) used with DIM and a *subscript* that indicates its place in the array.

```
B(1) B(2) B(3) B(4) B(5)
```

are the *subscripted variables* for the array dimensioned above.

An array variable name on the TS2068 can be either upper or lower case. Because the computer considers upper and lower case letters interchangeable when it comes to variables, you still have only 26 different single-letter variables.

Filling the Boxes

When an array is dimensioned, all its cells are automatically filled wtih zeros. Type in, as direct commands:

```
DIM A(2)
PRINT A(1), A(2)
```

You will get two zeros on the screen—the contents of the two "boxes."

Sharon Zardetto Aker, 20 Courtland Dr., Sussex, NJ 07461.

The great advantage of arrays over regular variables is the ease of assigning and retrieving values.

Placing a number into an array cell is like assigning a value to any other variable:

```
LET N(4)=17
```

might appear in a program, while

```
INPUT L(3)
```

would allow the cell to be filled while the program is running.

The subscripted variable that stands for an array element can be used in the same way that other variables are used, so:

```
LET A(4)=B(4)*C(2)
```

and

```
PLOT X(3),Y(3)
```

are valid programming statements.

Listing 1.

```
   5 REM "ARRAY DEMO"
  10 DIM A(10)
  15 FOR N=1 TO 10
  20 PRINT "WHAT DO YOU WANT IN
CELL ";N
  25 INPUT A(N)
  30 NEXT N
  35 CLS
  40 FOR N=1 TO 10
  45 PRINT "CELL ";N;" CONTAINS
";A(N)
  50 NEXT N
```

The great advantage of arrays over regular variables is the ease with which values can be assigned and retrieved. Listing 1 demonstrates this. It will ask you to INPUT a value for each of ten cells, and then print them out on the screen. Compare this program to one you would have to write if you were using regular variables instead; without the FOR-NEXT loops that can be used with subscripted variables, your listing would be about ten times longer.

An Application

Listing 2 is a short game that shows how you might use an array. It is a simple "guess my number" game with two added features: a subroutine that checks if you are repeating yourself when giving an answer, and a recap of all your guesses at the end. You have ten tries to find the correct number between one and fifty.

Lines 10-50 constitute the basic guessing game. The subroutine at line 100 checks the current guess (G) against the

Listing 2.

```
   5 REM "GUESS"
  10 DIM A(10)
  15 LET N=INT (RND*50)+1
  20 FOR T=1 TO 10
  25 SCROLL
  30 PRINT "TAKE A GUESS"
  35 INPUT G
  40 IF G=N THEN GOTO 55
  45 GOSUB 100
  50 NEXT T
  55 CLS
  60 PRINT ("RIGHT " AND G=N)+("
SORRY, " AND G<>N);"THE NUMBER W
AS ";N
  65 PRINT "YOUR GUESSES WERE:"
  70 FOR N=1 TO 10
  75 PRINT A(N)
  80 NEXT N
  85 STOP
 100 FOR K=1 TO 10
 105 IF G=A(K) THEN GOTO 130
 110 IF G<>A(K) THEN NEXT K
 115 LET A(T)=G
 120 RETURN
 130 SCROLL
 135 PRINT "YOU TRIED THAT ALREA
DY"
 140 GOTO 35
```

TS2068 Users: Delete lines 25 and 130.

contents of the array—each cell holds a prior guess. If a match is found, the player is informed, and the program returns to the INPUT line; if no match is found, the current guess is placed into the next available array cell. Lines 70-80 use a FOR-NEXT loop to print out all the wrong guesses.

Another Dimension

The "rows of boxes" in the previous programs are *single dimension* arrays, because you are working with a single row of boxes. If you visualize rows of boxes on top of each other or information presented in rows and columns, you will be dealing with a *two-dimensional* array.

A two-dimensional array is dimensioned by two numbers with the DIM statement, and each element in the array is identified by two subscripts.

```
DIM Z(3,4)
```

dimensions an array with twelve cells (three rows of four boxes). The first element is named Z(1,1) and the last Z(3,4). It is easy to keep track of the element names if you think of the first number as referring to a row and the second to a column. Figure 1 shows just how the cells in this array are named.

Figure 1.
Cell subscripts for a two-dimensional array.

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1	3,2	3,3	3,4

While the "stack of boxes" is a helpful image in understanding two-dimensional arrays, keep in mind that the computer does not actually have memory cells in that configuration.

An Application

Any data that benefits from a row/column presentation is a candidate for a two-dimension array. The program in Listing 3 uses a two-dimensional array. The program in Listing 3 uses a two-dimensional array to keep track of the number of local and long distance calls made during the course of a week.

The array is set up as two rows of seven cells. (It could have been seven rows of two cells.) The subroutine at line 50 fills the array with random data. Lines 20 and 25 set up the screen, lines 30-45 print the data on the screen.

The effectiveness of a two-dimensional array is not immediately apparent, because this program could have been written with a single dimension array, or even with separate variables for all the data (although assigning variable values would not have been so simple).

The importance of the two-dimensional array is evident when you want to manipulate the data. For instance, the total of local calls and long distance calls made during the week can be calculated with the following subroutine:

```
LET LOCAL=0
LET LONG=0
FOR N=1 TO 7
LET LOCAL=LOCAL+P(1,N)
LET LONG=LONG+P(2,N)
NEXT N
```

The first two lines of this subroutine are examples of variable *initialization.* Because we want to use LOCAL and LONG to store ever-increasing totals as the FOR-NEXT loop is performed, we use the statement

```
LET LONG=LONG+P(2,N)
```

and a similar one for LOCAL. Using these statements, however, without assigning *initial* values to the variables would result in an "undefined variable" error. This initialization procedure is used very often in games, where the variable that stores the score is initialized to zero. Array elements, remember, are automatically initialized to zero when you dimension the variable.

The latter part of this routine uses a FOR-NEXT loop to add the first column figures into LOCAL and the second column figures into LONG.

The average local or long distance calls per day are then LOCAL/7 and LONG/7. Try adding the printout of these statistics to your display, as well as writing a subroutine to calculate the total number of calls per day, and all the calls made during the week.

Other Dimensions

Numeric arrays are not limited to two dimensions. You might want to add statistics to the "telephone" program for each week of the year. In that case,

```
DIM P(52,7,2)
```

would set up a three-dimensional array, which you could visualize as 52 pages of rows and columns.

Listing 3.

```
   5 REM "CHART"
  10 DIM P(2,7)
  15 GOSUB 50
  20 PRINT TAB 14;"LOCAL";TAB 21
;"LONG"
  25 PRINT ,,"MON",,"TUES",,"WED
",,"THUR",,"FRI",,"SAT",,"SUN";A
T 2,0;
  30 FOR X=1 TO 7
  35 PRINT TAB 15;P(1,X);TAB 22;
P(2,X)
  40 NEXT X
  45 STOP
  50 FOR N=1 TO 2
  55 FOR X=1 TO 7
  60 LET R=INT (RND*20)
  65 LET P(N,X)=R
  70 NEXT X
  75 NEXT N
  80 RETURN
```

Arrays of more that three dimensions are possible, although progressively more difficult to visualize:

```
DIM(2,2,2,2,2,)
```

sets up a five-dimensional array with 32 elements (rows, columns, pages, books, shelves?).

The overall number of elements you can dimension depends on the memory you have available.

Caveats

Some computers will stop a program with a "re-dimensioning" error if you use the same array name twice. A ZX/TS computer will do what it is told: it will dimension the same array over again, wiping out all your data in the process.

So, be careful where you place your DIM statement—keep it outside of loops, as in the listings accompanying this article.

RUN will also wipe out the contents of an array, just as it erases regular variable values: use GOTO when you are working with a program with arrays!

String Arrays

String arrays are similar to numeric arrays in that a DIM statement is used to reserve memory space, and the use of subscripted variables makes the strings easy to store and retrieve.

However, string arrays have one major difference that calls for a more thorough study: for a string array, you have to indicate not only how many elements you want in the array (how many strings you want to store), but also how long the strings will be.

```
DIMN$(5,10)
```

will dimension an array that will hold five strings of ten characters each. If you use only one subscript when dimensioning a string array, e.g.,

```
DIMA$(10)
```

the computer will assume the number refers to the *length* of the string, and dimension an array of one element—and, yes, there are uses for a one-element string array!

While you can think of numeric arrays as rows of boxes, string arrays are more like a series of shelves, with only a certain amount of space on each shelf. Each string goes on one shelf, but each character of the string goes into one of the spaces on the shelf.

Once Upon a Time

This brings us to the story of Procrustes, a legendary (and, we hope, *mythical*) Greek innkeeper who forced travellers to fit one of two beds either by stretching their bodies to the right length, or by chopping off that portion of the anatomy that might be hanging over the edge.

What has that to do with string arrays? Well, assignment to a string array is *Procrustean:* a string will be either padded out with spaces to fill the array "shelf," or truncated to fit in. If we dimension a string array

```
DIMA$(3,6)
```

and put the strings *string, array,* and *storage* on the three shelves, they would be stored as shown in Figure 2: *string* fits perfectly, *array* gets a space added to it, and *storage* has its last letter removed. Listing 4 demonstrates what is illustrated in this figure. It will load the three words into an array and then print out the array contents. A period is added to the printout of each stored string so you can see exactly where it ends.

So What?

Why is it important that you take this Procrustean assignment into consideration? There are a number of reasons.

Dimensioning an array larger than you need uses up memory space. Saving

Figure 2. Procrustean string storage.

S	T	R	I	N	G
A	R	R	A	Y	
S	T	O	R	A	G

Listing 4.

```
 5 REM "STRING 1"
10 DIM A$(3,6)
15 LET A$(1)="STRING"
20 LET A$(2)="ARRAY"
25 LET A$(3)="STORAGE"
30 FOR N=1 TO 3
35 PRINT A$(N);"."
40 NEXT N
```

room for a dozen 30-character strings is a waste if you are going to store only five-letter words. More importantly, (yes, there are more important things than saving space), if you are going to be checking an INPUT against the contents of a string array cell, you had better be aware of just what is stored in that cell. Run the program in Listing 5 and type in the words as you are instructed.

Listing 5.

```
 5 REM "STRING 2"
10 DIM A$(2,5)
15 LET A$(1)="SHORT"
20 LET A$(2)="LONG"
25 FOR N=1 TO 2
30 PRINT "TYPE IN: ";A$(N)
35 INPUT B$
40 IF B$=A$(N) THEN PRINT "RIG
HT"
45 IF B$<>A$(N) THEN PRINT "WR
ONG"
50 NEXT N
```

When you type "SHORT" the computer will tell you that you are right, but, when you enter "LONG", you will be wrong.

Since the array was dimensioned to hold strings of five characters, it padded out "LONG" with an extra space at the end. If you type in LONG with a space after it, the computer will consider it a correct answer.

This aspect of Procrustean assignment can wreak havoc in a question/answer game if you do not allow for it. By dimensioning an array of one element to hold the answer (I told you there would be a use for it), the answer will be padded out to match the answer in the array. Add/change

```
 7 DIMN$(5)
35 INPUTN$
```

and run the program again.

Space Savers

If you are not sure just what dimensions to use in a program, try letting the user enter the figures that will be needed. For instance, at the start of an alphabet sort routine, you might use:

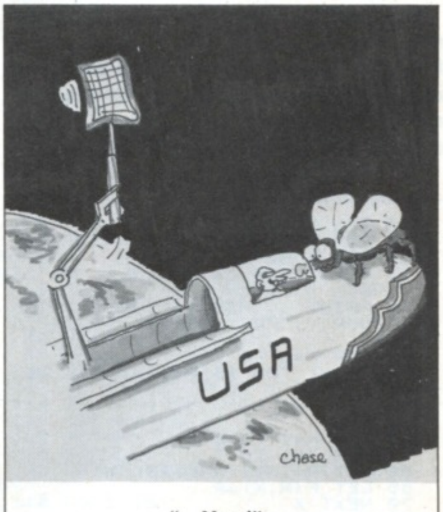
```
10 PRINT "HOW MANY WORDS?"
20 INPUT N
30 PRINT "LONGEST WORD?"
40 INPUT W$
50 DIMA$(N,LEN W$)
```

In this way, the array is kept to the exact number of elements needed, and the space in each cell is only as large as is necessary to store the longest word.

Another Application

Applications for string arrays are similar to those for numeric arrays in that they lend themselves to certain manipulations not possible with regular string variables. For instance, alphabetizing routines compare each string in an array with the others and rearrange their order until they have been sorted alphabetically; many word processing programs also use string arrays.

Another application that takes advantage of the Procrustean feature of ZX/TS string arrays is one that can save your display. If you have a game that includes a scoreboard, you may have room



"...Now!"

Use the same letter for both a number and a numeric array.

for only eight letters of the player's name. Asking a player to type in only eight letters is asking for trouble, since many people take perverse pleasure in not following directions. You can foil that person in one of two ways: check the length of the INPUT string and loop back if it is more than eight letters, or simply dimension of single-element string array to store the player's name. When it is printed out, only the first eight letters will be there.

Multi-Dimensional String Arrays

Setting up string arrays of more than one dimension is the same as setting up a multi-dimensional numeric array, except for the number that denotes the length of the strings.

```
N$(3,4,5)
```

dimensions a two-dimensional string array—three walls of four shelves with five spaces on each.

Bits and Pieces

Sinclair Basic has string-slicing functions that, while slightly non-standard, are easy to use, and they apply to strings inside arrays as well as to simple strings.

To pick a character out of an arrayed string, use

```
A$(3,4)
```

where 3 is the array element and 4 is the fourth character of the string stored there.

```
A$(3,2 TO)
A$(3,TO 5)
```

will give everything from the second character to the end of the string, and the first five characters of the string respectively.

These slicers can also be written as:

```
A$(3) (4)
A$(3) (2 TO)
A$(3) (TO 5)
```

What's in a Name?

The same letter can be used to name both a number and a numeric array in the same program; if you use a string variable name for a string array, it *cannot* be used for a simple string variable. You can use the same letter for up to three things in a program: a numeric variable, a numeric array, and a string variable (simple or array).

Onward . . .

Now that you know all about arrays, you should be ready to write such programs as word processors . . . mailing lists . . . spreadsheets . . . household budgets . . .

Crazybugs and Cyberzone

David and James Grosjean

Crazybugs is a Pac-Man type game, but with enough differences to make to make another enjoyable game. You are in a subterranean maze and are being chased by beetles. As the beetles lay eggs, one at a time, you must go, fetch, and lock up the egg in a flashing box. After locking up four eggs, you must exit the maze and move on to the next screen.

> **SYNC**
> **SOFTWARE PROFILE**
> **Name:** Crazybugs
> **Type:** Arcade maze
> **System:** TS2068
> **Format:** Cassette; cartridge
> **Summary:** Superb if you enjoy maze games and dislike cassette problems.
> **Price:** Cassette: $19.95
> Cartridge: $29.95
> **Manufacturer:**
> Timex Computer Corp.
> Waterbury, CT 06725

Crazybugs comes in two versions: cassette and Timex Command Cartridge. The cartridge very convenient because it avoids loading time. When the computer is turned on, the title screen for *Crazybugs* prompts the user to enter a skill level (1-3). If nothing is entered, the computer provides an entertaining demonstration.

When play commences, your humanoid appears in the lower left hand corner of the maze. You control him with a joystick or the Q, Z, I, and P keys. You must move quickly because the bugs are intelligent and will soon trap you if you remain idle. After you have locked up the four eggs, you must exit by a flashing white door on the maze's edge. Then the screen flashes colors, the computer makes unusual sound effects, and the maze undergoes a metamorphasis as a flashing orange box moves around closing up branches of the maze to set up the next screen.

The second screen is harder because there are fewer places to run. Luckily, you are faster than the bugs and have five lives. The third screen is the hardest. It completes the first phase.

The second phase begins with the original maze but has an added twist. This screen has four beetles, plus one super-bug. which is much faster than you. Luckily, it is not so smart, and it stops before turning a corner. Another super-bug is added each phase.

This game has many fine points for which it must be commended. 1) The cartridge avoids loading time and allows starting a new game by switching the computer off and on again. 2) *Crazybugs* makes excellent use of the TS2068's hi-resolution graphics, color, and sound. 3) The game is progressively difficult, providing continuous challenge. 4) Because of the simple concept of *Crazybugs*, you do not have to read endless pages of directions. 5) The game has on-screen scoring and a hi-score indicator.

As for disadvantages, *Crazybugs* has only two: 1) initials cannot be entered when a hi-score is achieved; 2) the game has no pause feature, as many games now do. Thus, this game is superb for people who enjoy maze games and who dislike cassette problems.

David and James Grosjean, 50 Kings Rd., Chatham, NJ 07928

Cyberzone, the new, improved, and updated version of *TS Destroyer* (originally published in *SYNC* 2:4), takes advantage of the sound and color graphics of the TS2068. The object is to shoot through the rotating defense shields and destroy the alien inside. *Cyberzone* adds some excellent and challenging features.

> **SYNC**
> **SOFTWARE PROFILE**
> **Name:** Cyberzone
> **Type:** Arcade space
> **System:** TS2068
> **Format:** Cassette
> **Summary:** If you liked *TS Destroyer*, a must; one of the best new games for the TS2068.
> **Price:** $19.95
> **Manufacturer:**
> Softsync, Inc.
> 14 E. 34th St.
> New York, NY 10016

When the program loads, you have the option of intructions, play, or attract mode. If you leave the title page alone for about 20 seconds, the computer automatically goes into attract mode. To play, simply choose a skill level (1-5). Controls move left and right (arrow keys) and 0 to fire. A joystick can be used instead.

At first, the game seems to be merely an excellent translation of the old game, but soon an alien plane appears and you know that this is different. The plane emerges at the top and swoops down, taking a seemingly unpredictable and unorthodox path to drop deadly accurate bombs. They do not make any kamikaze moves, but, when they are shot down, they fall from the sky like a sack of wet cement.

Just as you find that the planes and very accurate laser mounted the alien's underside are about your match, suddenly, from the center of the alien's ship, a wide beam transports a flying saucer-looking object down to a level just above your own. It begins to shoot down your shots, and you cannot blow it away. Then it starts to move and shoot down your shots everywhere you go. You can only destroy it by getting one of the other enemies on the screen to destroy it for you.

If you eventually get through the defenses and hit the alien, there is a bloody explosion and you are advanced to a harder screen. In the later screens, two planes attack you, the laser gets faster, and everything gets tougher.

This game is an excellent job of programming and a source of constant challenge and excitement. The (optional) sound effects, colorful hi-res graphics, multiple levels of play, and high speed action make this a fun game that can be enjoyed by anyone. Small features such as a record of the hi-score, attract mode, interesting destructions, and optional sound complete its engrossing effect.

If you liked *TS Destroyer*, this version is a must. *Cyberzone* is one of the best new games for the TS2068.

Whither Clive?

David H. Ahl

This is a somewhat roundabout story, but bear with us. Lotus, that outstanding manufacturer of winning race cars as well as the Lotus Elite and Elan, has been in serious financial difficulties lately. These troubles have been magnified by the death of Colin Chapman, the founder of Lotus, and by the DeLorean affair. The only reason that DeLorean enters into the picture is that Lotus had agreed to use the DeLorean plant for the development and manufacture of some very sophisticated lightweight cars.

The latest reports we get indicate that things are improving rapidly. First, Lotus has entered into a cooperative sports car project using engine and drivetrain components from Toyota. This sparked some spirited negotiations on the part of David Wickins of British Car Auctions to take over Lotus before it fell to the control of (horrors!) a Japanese firm. The Lotus board accepted Wickin's offer.

However, some weeks later Toyota made an offer to buy 2.9 million shares of Lotus stock (16% of the total) for £1.16 million. This offer was accepted at the Lotus stockholders' meeting. So where does Sir Clive Sinclair enter in?

Sir Clive, that electronics wizard who invented the "black watch," cheap scientific calculator, ZX80 (and its derivatives), and flat screen TV, had a new invention on the drawing boards—an electric car (shades of Tom Swift).

So he took a small dollop of his vast wealth (£9 million) and formed a new company called the Sinclair Vehicle Project and bought the DeLorean plant in Ireland. Why? Because Lotus had already modified it to build sophisticated lightweight vehicles.

Sinclair has stated his theory that, if you can't lighten the batteries (apparently you can't), you should 1) make the batteries fit the vehicle and/or 2) make the vehicle more efficient. His method for doing the latter is to use a chip (he calls it a "chopper") to chop the vehicle power demand on the battery into time slices of milli- or nanoseconds. In other words, deliver one millisecond of power then rest for two. Apparently this extends the discharge time enormously. Also, with the chip, the system can be programmed for various types and weights of vehicles.

Sounds good to us. Now when do we get one for evaluation? —*DHA*

Knight's Tour

Robert J. Midura

The Knight's Tour Puzzle

The "Knight's Tour" is a familiar chess puzzle. The object is to move a knight according to chess rules around the board in such a way that each square is visited only once. The puzzle may be solved by using a trial and error process that systematically explores all of the moves until a solution is found.

The program in Listing 1 will allow you to find a solution to the puzzle with your computer. You set the size of the board, pick the knight's starting point, and sit back and watch the knight make its tour of the board.

The Programming Algorithm

The program uses what is called a backtracking algorithm to arrive at a solution. The board locations of each successful move are stored on a stack along with the direction that allowed a move to proceed to another. When a position is found from which no move can be made, the program pops off the stack and continues with the next direction that was stored with the move. In this way the program remembers where it left off trying new moves at every location.

The board is displayed and the moves are entered on the board with the current location of the knight shown in inverse. This enables you to observe the backtracking process on the screen.

Entering the Program

1) Before typing in the program, we need to set up an array to store the values used to compute moves from a location on the board. Since a knight has 8 possible moves from a given location, we need an 8 x 2 array. In the immediate mode type in DIM A(8,2) and ENTER.

Robert J. Midura, 19 Merrifield St., Worcester, MA 01605.

Successful moves are stored on a stack so the program remembers where it left off trying new moves at every location.

2) Then type in the short program in Listing 2.

3) Start the array input program by typing GOTO 10 and enter the values shown in Table 1.

Table 1. Array values.

1	-1	-2
2	-2	-1
3	-2	1
4	-1	2
5	1	2
6	2	1
7	2	-1
8	1	-2

4) Type in Listing 1 which will erase the program you used to input the array values.

Running the Program

1) Put the computer in SLOW mode and type GOTO 900 and ENTER.

2) The program will first request the size of the board. It will accept only values from 1 to 8. I have found that boards smaller than 5 x 5 are unsolvable. Of course, a 1 square board is solved simply by one move. (Try starting with a 5.)

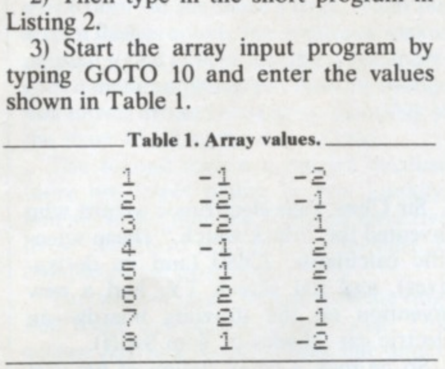
3) The program will request the row and column numbers for the location of the starting point of the knight. (Try using row 5 and column 1.)

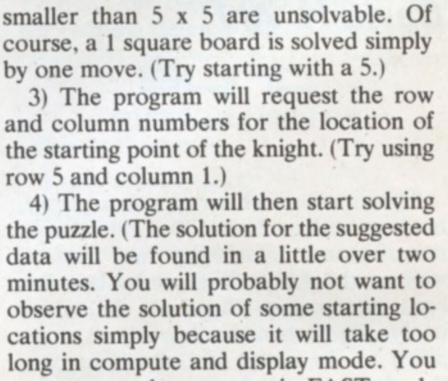
4) The program will then start solving the puzzle. (The solution for the suggested data will be found in a little over two minutes. You will probably not want to observe the solution of some starting locations simply because it will take too long in compute and display mode. You can operate the program in FAST mode by first entering the line

225 SLOW

so the screen will activate when a solution is found. Then put the computer in FAST and type GOTO 900)

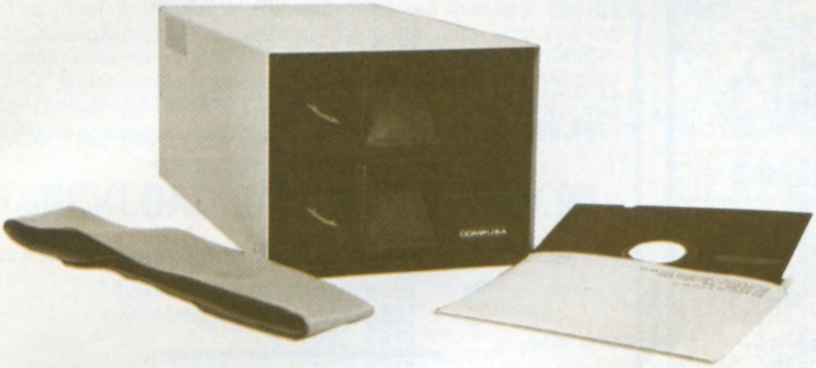
5) The program will then give you the option of finding another solution from the same input or of resetting the board





New from COMPUSA

FDC-100 SERIES FLOPPY DISK SYSTEM for TIMEX SINCLAIR USERS



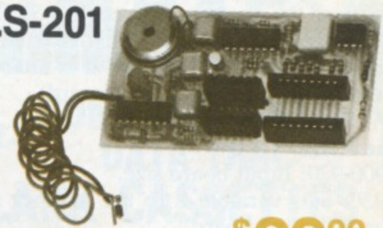
Available in both single disk and double disk configuration

- Up to 250 K bytes per side
- High speed up to 31.2 K character/sec
- SAVE and LOAD from Timex/Sinclair BASIC no PEEK's, POKE's or USR's
- All commands fully integrated into Timex/Sinclair Basic and do not interfere with tape SAVE and LOAD
- System compatible with other computers
- Extensive command list
- Create your own data types via READ and WRITE physical track commands

$499 (price for FDC-101 single drive system)

Coming Soon
COLOR SIN 81
WATCH OUR ADS FOR DETAILS

BLEEPER SYSTEM — BLS-201



$29.00

- Positive signal for every key press on your ZX81, T/S 1000.
- The *BLEEP* you hear provides positive audio feedback that a key has been depressed and makes data entry/program entry much easier.
- BLS-201 is easily installed in minutes and comes with complete, easy to follow instructions.

(SPECIAL: Order with LOS-200 Tape Loading System and pay only $14.00 save over 50%).

TAPE LOADING SYSTEM LOS-200

Eliminate the frustration of ZX81 and T/S 1000 tapes and commercial tapes that won't load. The LOS 200 provides the facility to check volume and blank tape noise and to determine if a tape is damaged.

$36.50

FEATURES:

- LED indicators show: volume too high or low-noisey tape end of program-"voice over".
- Earphone plug allows you to listen to the tape while setting up and while loading tape. You can hear drop outs and bad tone setting quite easily.
- Reset switch allows starting computer from "K" without removing power plug.
- SAVE/LOAD switch allows both EAR and MIKE to be plugged into computer at once. You may simply switch between SAVE and LOAD.
- Powered completely by the computer and does not interfere with the operation of any other peripheral.

Send me more information on

☐ FDC 100 Series ☐ BLS 201 ☐ LOS 200 ☐ Color SIN 81
☐ My check is enclosed ☐ Master Card ☐ Visa ☐ COD $3 charge

Please ship:
☐ FDC 101 ☐ FDC 102 ☐ BLS 201 ☐ LOS 200

Card No. ____________ Expiration Date ________ Signature ________

NAME: ____________________

ADDRESS: ____________________

CITY: ____________ STATE ________ ZIP ________

TELEPHONE: (_____) ____________

Add $3.50 ea. item for shipping/handling. NJ Residents add 6% Sales Tax.

COMPUSA

COMPUSA CORPORATION
1101 Bristol Road
Mountainside, New Jersey 07092 U.S.A.
(201) 654-7220 Telex: 138275

CIRCLE 7 ON READER SERVICE CARD

size and starting point. (If you choose to try for another solution with the suggested input, you will get one in about 3 1/2 minutes. The program will start back tracking from the solution it just found in an attempt to find another.)

6) Pressing any key other than A or R will cause the program to execute a STOP statement. The program will also stop and leave the board blank if it is unable to find a solution (try a 3 x 3 board).

Line Notes

900-920: Input board size.

930: Sets variable E to the number of squares on the board.

940: Sets X to the string slice length for printing the board.

950: Clears the screen.

1000-1050: Print the board.

1060: Defines stack array; size set to number of squares by 3 so that it can store the row, column, and direction value for every move made. The direction value is actually the index to the array A indicating which values are being used to compute trial moves.

1070: Defines a string array to represent the board and to check if a square is already occupied. A blank indicates that the square is unoccupied; an X that the square is occupied.

1080: Sets variable P to point to top of the stack. Since your starting location is the first move, it is stored at S(1,1) and S(1,2). So P begins at 2.

1090: Sets variable C to point to the current move in the stack which is 1 to begin with.

1100-1165: Input the starting location of the knight.

1170: Prints knight at this location.

1180: Marks board position.

1190: Sends control to main part of program (lines 10-220).

10: Sets up trial move loop. The FOR-NEXT variable I is used as the index to the array A.

20, 30: Compute a possible move using the values in A and the current location of the knight. I takes on the values 1 to 8 to examine all the moves until one is found from the current location.

40: Checks if the move is on the board and if the square has not been visited. When a move cannot be made from a location after all 8 moves have been tried, the program proceeds past the NEXT (line 50). If all the conditions in line 40 are met, the program branches to 150.

60, 70: Blank out the current location on the screen and board map.

80: Backs up to previous location.

100: Prints it in inverse.

110: Decrements P so that the next move found erases the deadend move.

120: Sets I to the last direction tried at this previous location so the NEXT in line 130 will set I to point to the values in

Listing 1. Knight's Tour Basic program.

```
  10 FOR I=1 TO 8
  20 LET X=S(C,1)+A(I,1)
  30 LET Y=S(C,2)+A(I,2)
  40 IF X>0 AND Y>0 AND X<=L AND
 Y<=L THEN IF B$(X,Y)=" " THEN G
OTO 150
  50 NEXT I
  60 PRINT AT S(C,1)*2,S(C,2)*3;
"  "
  70 LET B$(S(C,1),S(C,2))=" "
  80 LET C=C-1
  90 IF NOT C THEN STOP
 100 PRINT AT S(C,1)*2,S(C,2)*3;
(CHR$ (INT (C/10)+156) AND C>9);
CHR$ (C-INT (C/10)*10+156)
 110 LET P=P-1
 120 LET I=S(C,3)
 130 NEXT I
 140 GOTO 60
 150 PRINT AT S(C,1)*2,S(C,2)*3;
C;AT X*2,Y*3;(CHR$ (INT (P/10)+1
56) AND P>9);CHR$ (P-INT (P/10)*
10+156)
 160 LET S(P,1)=X
 170 LET S(P,2)=Y
 180 LET S(C,3)=I
 190 LET C=P
 200 LET B$(X,Y)="X"
 210 LET P=P+1
 220 IF P<=E THEN GOTO 10
 225 SLOW
 300 PRINT AT 21,0;"HIT A=ANOTHE
R SOLUTION,R=RESTART"
 310 LET I$=INKEY$
 320 IF I$="" THEN GOTO 310
 330 PRINT AT 21,0;"
                              "
 340 IF I$="A" THEN GOTO 10
 350 IF I$="R" THEN GOTO 800
 360 STOP
 800 CLS
 900 PRINT "ENTER SIZE OF BOARD"
 910 INPUT L
 920 IF L<1 OR L>8 THEN GOTO 910
 930 LET E=L*L
 940 LET X=L*3+1
 950 CLS
1000 PRINT "   1  2  3  4  5  6
 7  8"( TO X+1)
1010 FOR I=1 TO L
1020 PRINT TAB 2;"+--+--+--+--+-
-+--+--+--+"( TO X)
1030 PRINT I;TAB 2;"+  +  +  +
+  +  +  +  +"( TO X)
1040 NEXT I
1050 PRINT TAB 2;"+--+--+--+--+-
-+--+--+--+"( TO X)
1060 DIM S(E,3)
1070 DIM B$(L,L)
1080 LET P=2
1090 LET C=1
1100 PRINT "ENTER STARTING LOCAT
ION"
1110 FOR I=1 TO 2
1120 PRINT AT I+L*2+2,0;"ENTER "
;("ROW" AND I=1)+("COLUMN" AND I
=2)
1130 INPUT X
1140 IF X<1 OR X>L THEN GOTO 113
0
1145 PRINT AT I+L*2+2,0;"          ";
TAB 14;X
1150 LET S(C,I)=X
1160 NEXT I
1165 PRINT AT L*2+2,0;"          "
1170 PRINT AT S(C,1)*2,S(C,2)*3;
"█"
1180 LET B$(S(C,1),S(C,2))="X"
1190 GOTO 10
```

TS2068 Adaptation

To use this program on the TS2068, type in the listing with the following line changes.

```
 100 INVERSE 1: PRINT AT s(c,1)*
2,s(c,2)*3;c: INVERSE 0: PAUSE 2
0
 150 PRINT AT s(c,1)*2,s(c,2)*3;
c: IF p<e THEN INVERSE 1: PRINT
AT x*2,y*3;p: INVERSE 0: PAUSE 2
0
 155 IF p=e THEN PRINT AT x*2,y*
3;p
255 (delete)
```

A that will produce a move in the next direction.

140: GOTO instructs program to back up if all the moves at this point have been exhausted until a location can be found from which a not previously tried move may be made.

150: Prints current location in normal video and new location in inverse.

160, 170: Store the new location in the stack at P.

180: Stores the index to A that was used to compute the new location with the old location data in the stack at C.

190: Sets current move pointer, C, to point at the new location data in the stack.

200: Marks the board map.

210: Increments pointer P to point to the next free space in the stack.

220: Checks if the board is full; indicated by the stack being full and, if so, the solution is found. If not, the program branches to the beginning of the trial move loop.

300-360: Restart option if solution is found. ■

Listing 2. Array input program.

```
10 FOR I=1 TO 8
20 PRINT I;",";1
30 INPUT A(I,1)
40 CLS
50 PRINT I;",";2
60 INPUT A(I,2)
70 CLS
80 NEXT I
```

"...This species doesn't bloom, it just belches..."

ZX PRO/FILE CARTRIDGE WITH "QUICKLOAD"! $39.95

The most professional file and database manager for the TIMEX—now on plug-in cartridge! Type one command and the program runs instantly. But that's not all—you also get "QUICK-LOAD" which saves and loads 16K in just 30 seconds! With 59 page manual! SATISFACTION GUARANTEED.

BOTH on one plug-in cartridge...................... **$39.95**

QUICKLOAD CARTRIDGE $29.95

- Saves/loads 16K in 30 seconds
- Stores over a megabyte on one cassette
- Next best thing to a disk drive

- Check or money order
- Shipping add $2
- Cal. res. add 6½ %
- SASE for catalogue

Both cartridges have rear expansion port for 16/64K RAM.

For the **LARGEST** selection of EPROM software

Rompak TS1000 1500

8206 Blackburn Ave.
Los Angeles, CA 90048

STOP

ARE YOU JUST A LITTLE TIRED OF REFLEX ACTION GAMES? TRY OUR MINDWARE BY ISTHMUS

TRIVIA CHALLENGE: the new innovative software package. **TRIVIA CHALLENGE** is the trivia game for adults which tests your knowledge in a variety of categories and provides countless hours of enjoyment. Its fun, challenging, educational and can be enjoyed by 2 to 8 players. This one of a kind software package comes to you on four 92 minute cassettes for the unbelievable low price of only $18.95 plus $1.95 S&H $18.95

— No shipping and handling charge on following cassettes —

MASTER MIND: An intriguing strategy game. Break the codes. For 1 or 2 players $7.95

BATTLESHIP: A naval strategy game for 2 players $7.95

SPELLER QUIZ: Ideal program for children to enter their entire spelling lessons, lists, spells, and then tests each word and retests for 100% efficiency. Makes learning fun. For 1 or 2 students $6.95

ANAGRAMS: A timed word puzzle for 2 to 4 players. All ages. $5.95

HANG ROBOT: A unique variation of an old favorite. 1 to 4 players enter their own words or solve from the pre-programmed words $5.95

TIC-TAC-TOE: Even a game for pre-schoolers, fun for children as they learn about computers $5.95

Send Check or Money Order to:

ISTHMUS
P.O. Box 174
Winnipeg, Manitoba
Canada R2M 4A5

CIRCLE 30 ON READER SERVICE CARD

YORK 10™ CASSETTES

COMPUTER GRADE DATA TRAC BLANK CASSETTES

C-05, C-06, C-10, C-12, C-20, C-24, C-32

From the leading supplier of Computer Cassettes, new, longer length C-12's (6 minutes per side) provide the extra few feet needed for some 16K programs.

- BASF-LHD (DPS) world standard tape. Premium 5 screw shell with leader.
- Internationally acclaimed. Thousands of repeat users.
- Error Free • Money back guarantee.

500 C-12's or C-10's — 38¢ each
(w/labels, add 4¢ • Shipping $17./500
500 Boxes 13¢ ea. • Shipping $10./500
(Free Caddy offer does not apply)

NEW

TRACTOR FEED • DIE-CUT BLANK CASSETTE LABELS
WHITE: $3.00/100 $20.00/1000
COLORED LABELS • Pastels - Red, Blue, Green, Yellow, Lavender
$4.00/100 $30.00/1000

CASSETTE STORAGE CADDY
Holds 12 cassettes w/o boxes
Includes edge labels and index card
$2.95

FREE
1 CADDY WITH EVERY 4 DOZ. CASSETTES PURCHASED
(does not apply to 500 quantity offer)

VISA **Call: 213/700-0330** for IMMEDIATE SHIPMENT on Credit Card Orders

ORDER NOW... MAIL TO... **YORK 10™** 9525 Vassar Ave. #S Chatsworth, CA 91311

ORDER FORM

ITEM	1 DOZEN	2 DOZEN	TOTAL
C-05	☐ 7.00	☐ 13.00	
C-06	☐ 7.00	☐ 13.00	
C-10	☐ 7.50	☐ 14.00	
C-12	☐ 7.50	☐ 14.00	
C-20	☐ 8.75	☐ 16.50	
C-24	☐ 9.00	☐ 17.00	
C-32	☐ 11.00	☐ 21.00	
Hard Box	☐ 2.50	☐ 4.00	
White Labels	☐ 3.00/100	☐ 20.00/1000	
Colored Labels Color ____	☐ 4.00/100	☐ 30.00/1000	
DESCRIPTION	PRICE	QUANTITY	
Storage Caddy	2.95		
		SUB TOTAL	
Calif. residents add sales tax			
Shipping/handling (any quantity — using prices above)			3.50
Outside 48 Continental States — Additional $1 per caddy; per doz. cassettes or boxes.			
		TOTAL	

#S

Each cassette includes two YORK 10 labels only. Boxes are sold separately. We prefer to ship by UPS as being the fastest and safest. If you need shipment by Parcel Post, check here ☐.

NOTE: Additional charges outside 48 Continental States. Shipments to AK, HI, and USA possessions go by Priority Mail. Canada & Mexico—Airmail. All others— Sea Mail.

Ask about our DUPLICATING SERVICE

CHECK OR M.O. ENCLOSED ☐ Charge to Credit Card: VISA ☐ MASTERCARD ☐

☐ PLEASE SEND QUANTITY DISCOUNTS

Card No. ______________________ Exp. ______

Name ______________________________

Address ______________________________

City ______________ State/Zip ______________

Signature ______________ Phone ______________

Computer make & model ______________ Disk? (y/n) ____

CIRCLE 68 ON READER SERVICE CARD

Meteors

Michael B. Williams

The Problem

You are piloting a spaceship when suddenly appears an onslaught of meteorites which must be avoided at any cost. Your warp system is malfunctioning and can be used, but its duration varies. The situation seems hopeless!

Sound challenging? How would you like to try to save the ship on the unexpanded ZX81? It may sound impossible, but it is not. Here is a technique that will allow arcade-style games on the ZX81—even in 1K.

The Solution

Often clouded in mystery is the display file—the place in memory which stores the picture you see on your screen. In general, it is 24 lines by 33 characters (32 columns + ENTER) and occupies 793 bytes of memory. The address of the display file is held in two bytes with a special name: D-FILE. The first character in the display file is always ENTER. This is why it is 793 bytes long and not 792. The address of any position in the display file can be calculated by finding its start, adding 33 for each line down it is, and adding its column number. Somewhat tedious, isn't it?

Notice that I said, "in general." This is because, when the computer finds it has less than 3¼K RAM available (according to RAMTOP, another system variable), the display file is left with just ENTERs to serve as an end-of-line character. Thus, the system is thoughtfully designed so that each of the 24 lines can contain a different number of characters. How, then, can you calculate the address of a certain screen position? The truth is, you do not even need to.

The PRINT AT subroutine in the ROM is required to find the address of a particular position held in registers B (line) and C (column) and set the PRINT position there. The current PRINT position is held as two bytes in DF-CC. These are the bytes the command PRINT AT will modify.

Often clouded in mystery is the display file—the place in memory which stores the picture you see on your screen.

To find the character at a certain position, you PRINT AT the position and PEEK the address contained in DF-CC. This technique is illustrated in lines 60 and 70 of Listing 1. That is all there is to it; you do not even need a full display file to do memory mapping.

Do not be surprised to find an ENTER character (code 118) at the position. This simply indicates there is nothing there yet. What CLS does is to look at RAMTOP to determine if there is sufficient memory to fill the screen with spaces. If not, it just places 25 ENTERs in the display file. There must always be these 25 ENTER characters present. POKEing an ENTER character could very well crash the system.

So our fantasy of dodging meteors hurtling through space is now within reach. Examine Listing 1. Once you understand how it works, you can create fast-moving games in just 1K RAM.

The first thing you should notice is how "outer space" is printed. I could have used a FOR-NEXT loop, but the

Table 1. Important address/system variables.

Address	# bytes	Name	Description
16388	2	RAMTOP	Stores address of first nonexistent byte; varies with amount of RAM available.
16396	2	D-FILE	Start of display file; first character is always ENTER. Warning: *Never* POKE an ENTER.
16398	2	DF-CC	Holds current PRINT position; PEEK after PRINT AT to find character at that position.
16442	1	S-POSN	Holds number of lines left on screen; line number is equivalent to 24-PEEK 16442.
16418	1	DF-SZ	Number of lines on bottom of screen; can be POKEd to zero to use all 24 lines. Warning: Do not use SCROLL or INPUT if value is less than 2.
08F5h		PRINT AT	8K monitor routine to set PRINT position as held in registers B (row) and C (col). Useful for machine language programs.

Michael B. Williams, 1300 DePaul Way, Virginia Beach, VA 23464.

method used here is more economical and just as fast. The bytes called S-POSN store the number of positions to the left of and including the current position and the number of lines below and including the current PRINT position (see Table 1 for a list of all bytes referred to). The second byte of S-POSN is useful for determining when to use CLS to avoid the out-of-screen error. If you add a line

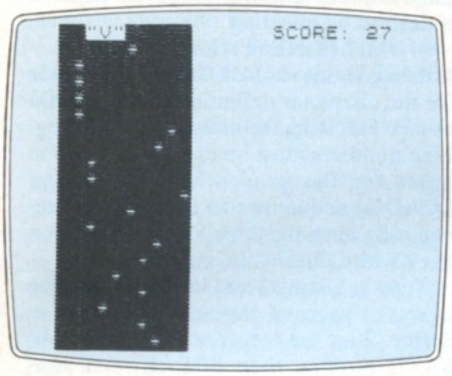
```
IF PEEK 16442=2 THEN CLS
```

you will not run out of screen and have to use CONT. You can see how this technique was used in line 20.

Lastly, the address at 16418 (called DF-SZ) holds the number of lines at the bottom of the screen. If this is POKEd to zero, you can use all 24 lines! Whenever any report code is given, though, its value will return to two.

Also, there is a warning when POKEing DF-SZ: if its value is less than two, SCROLLing or INPUTing will crash the system (if you do not know what that is, try it and see what happens).

So POKEing this will not do much good in our program, but it will remove the 22-line restriction and allow a greater amount of data to be shown on the screen at one time.



Listing 1.

```
 10 PRINT "          "
 20 IF PEEK 16442<>2 THEN RUN
 30 LET P=4
 40 LET S=0
 50 LET W=5
 60 PRINT AT 0,0;
 70 LET Z=PEEK 16398+256*PEEK 1
6399
 80 IF NOT W AND PEEK (Z+P)+PEE
K (Z+P+1)+PEEK (Z+P+2)<>384 THEN
 GOTO 170
 90 IF NOT W THEN PRINT AT 0,P;
""V""
100 LET S=S+1
110 IF INKEY$="0" THEN LET W=1
120 IF RND>RND THEN LET W=0
130 LET P=P-(INKEY$="5" AND P)+
(INKEY$="8" AND P<7)
140 SCROLL
150 PRINT "          ";AT 21,9*
RND;"*"
160 GOTO 80
170 PRINT AT 0,16;"SCORE: "+STR
$ S
180 PRINT AT 0,P;"""V""";AT 0,P
;"""V"""
190 IF INKEY$<>"R" THEN GOTO 18
0
200 PRINT AT 0,0;
210 RUN
```

Meteors

Type in the program as listed using the graphics notes. To begin, make sure you are in SLOW mode; type RUN and ENTER. The display represents space. You avoid the meteors by pressing the 8 key to move left and the 5 key to move right. Pressing the 0 key activates the warp, which makes your ship invulnerable to the meteors, but lasts only a random amount of time. The game is over when any part of your spaceship collides with a meteor. After your score is displayed, press R to play again.

Program Notes

10: Inverse space (10).

90: Inverse quotation marks; inverse V; inverse quotation marks.

150: Inverse space (10); inverse asterisk.

180: Double quotation marks (on Q key); double quotation marks; inverse quotation marks; inverse V; inverse quotation marks.

These are just a handful of ideas for which the display file may be used. If you have discovered some others, drop us a line.

Defining Characters

Robert D. Hartung

Among the extended capabilities of the TS2068 are its high resolution graphics and user defined graphics characters (UDG). Since the definition of each character requires 64 binary inputs, these entries can be made much more easily if we set up a little more elaborate input routine than the one given in the manual.

We will consider first a routine to define and store 21 characters, the limit of the UDG file. Then we will look at one method of providing for the definition, storage, and recall of as many "pages" or sets of characters as we have the inclination and RAM capacity to define.

Character Definition Routine

Listing 1 provides one routine for entering the characters more easily. It starts with line 9000 to permit MERGEing it with other programs if desired.

Type in the program, RUN, and ENTER.

12345678 INPUT e FOR ERROR

will appear. You are ready to draw your own character by using the 1 and 0 keys as your pencil and paper. Pressing 1 gives a black block; 0, a yellow block. If you make an error, press e and start over. The computer assigns your character to the next open letter.

When the twenty-first character is completed, the whole "page" of characters is displayed. BREAK may be used at any time, followed by GO TO 9600 to revise the character set or GO TO 9700 to SAVE to tape.

To SAVE to tape, type GO TO 9700 and start the recorder at the first prompt. Continue recording and press any key at the second prompt. BREAK to get out of LOAD mode after both SAVEs have been completed. CAPS SHIFT 9 for GRAPHICS mode will then produce the new character set on keys *a* through *u*.

Additional Pages

To provide for storing more than one set of 21 defined characters or "pages" in memory above RAMTOP, we must first CLEAR the required area. Each set requires 168 bytes of storage plus 12 bytes for the LDIR block transfer routine which transfers sets of data back and forth between the storage area and the UDG area beginning at 65368 in RAM. Since CLEAR deletes all variables, we will preface the LOADing of the main program with the short routine in Listing 2 which stores in line 1 REM nn the number of pages to be CLEARed in memory, CLEARs this RAM area, then goes automatically into MERGE mode for the main program following it on tape. The second n byte in 1 REM may be used to store a starting page number safe from RUN or CLEAR if desired by inserting

9312 POKE (VAL r$+6),pno

It then goes into LOAD "bin" CODE mode for the character definition codes SAVEd to tape following the main program listing. Line numbers must be exactly as listed in order for the proper MERGEing and LOADing sequences to take place automatically when the three SAVEs are played back with LOAD "bin" or LOAD "".

Listing 1. Character definition routine.

```
   1 REM ??
9000 LET r=97
9004 POKE 23692,-1
9005 PRINT AT 21,0
9007 PRINT "12345678  INPUT e FO
R ERROR"
9008 FOR g=r TO 117
9009 FOR i=1 TO 8
9010 LET byte=0
9015 LET b$="BIN "
9020 FOR n=7 TO 0 STEP -1
9030 PAUSE 0
9035 LET a$=INKEY$
9040 IF a$="e" THEN LET r=g: GO
TO 9005
9050 IF a$<>"1" AND a$<>"0" THEN
 GO TO 9030
9055 LET b$=b$+a$
9060 LET byte=byte+2↑n*VAL a$
9070 IF a$="0" THEN INK 6: PRINT
 "█";: INK 0
9080 IF a$="1" THEN PRINT "█";
9090 NEXT n
9100 PRINT i;TAB 10;b$;TAB 24;by
te
9110 POKE USR CHR$ g+i-1,VAL b$
9130 NEXT i
9140 PRINT CHR$ g;TAB 9;CHR$ (g+
47)
9150 NEXT g
9600 PRINT "CHR$ PRINT OUT "
9610 FOR n=144 TO 164
9620 PRINT CHR$ n;
9630 NEXT n
9640 PRINT '"abcdefghijklmnopqrs
tu"
9650 PRINT '''" INPUT NEW CHR$ A
T (a TO u)"'" INPUT s TO SAVE CO
DE "
9660 PAUSE 0
9680 LET r=CODE INKEY$
9685 IF a$="s" THEN GO TO 9710
9690 GO TO 9004
9700 SAVE "bin" LINE 9720
9710 SAVE "bin"CODE 65368,168
9720 LOAD "bin"CODE 65368,168
```

Listing 2. Short clear memory routine.

```
   1 REM ?n
9002 LET r$="PEEK 23635+PEEK 236
36*256"
9010 PRINT "NO. PAGES TO CLEAR ?
"
9020 INPUT nop
9030 POKE (VAL r$+5),nop
9040 SAVE "bin" LINE 9050
9050 CLEAR 65356-168*nop
9060 LET r$="PEEK 23635+PEEK 236
36*256"
9070 LET nop=PEEK (VAL r$+5)
9240 MERGE "bin"
9245 LOAD "bin"CODE 65356-168*no
p,168*nop
```

Type in Listing 2 and input the number of sets or pages of characters you wish to define. Start the recorder at the line 9040 SAVE prompt, then BREAK, and stop the tape when the LOAD mode commences. Enter LOAD "bin" and reLOAD from tape the character definition program in Listing 1. (You did SAVE it, didn't you?)

Add to it the lines and changes indicated in Listing 3.

SAVE the listing to tape immediately following the short CLEAR memory routine by using GO TO 9200. Now rewind the tape, enter LOAD "bin" or LOAD "", and start the tape at the CLEAR memory routine. Let it run until both it and the main listing are reLOADed. BREAK and GO TO 9300. Begin defining your characters at PAGE 1. At any point before the completion of a page or set of 21 characters, entering "a" will return you to the printout and menu where entering "s" will start the SAVE of the definition codes to tape, immediately following the SAVE of the main listing. All three SAVEs will LOAD one after the other by starting with LOAD "bin" or LOAD "".

It is not necessary to MERGE the entire character definition listing with a program in which you wish to use various sets of characters. First, make a SAVE to tape of the short CLEAR memory routine. Next, reLOAD the main character definition routine and insert line 9500 RETURN. Delete all lines except 1 REM nn up to 9200 and delete all lines after 9500 except 9999. Delete lines 9245, 9335, 9340, 9410, 9470, and all DATA in 9480 following the first 201. The second set of DATA is no

Robert D. Hartung, PO Box 125, Palmyra, NY 14522.

longer needed since all block transfers will now be made only from the page files to the UDG file.

When editing has been completed, enter as a direct command

LOAD "bin" CODE 65356-168*nop,16
8*nop

and LOAD the definition codes (the third SAVE in the original series). Immediately following the CLEAR memory routine on the tape, SAVE the abbreviated listing and the codes by GO TO 9200, then BREAK, then GO TO 9999.

These three consecutive SAVEs will reLOAD in sequence by starting with a LOAD "bin" or LOAD "". They may now be MERGEd with a program using UDG characters as long as it does not contain a line 1 or any other line numbers used in the UDG subroutine.

A typical program line to call up a new page of UDG characters would be like this

10 LET pno=1: GO SUB 9315: PRI
NT "a"

The letter "a" is entered in GRAPHIC mode (CAPS SHIFT 9) and "pno" is the page number. Depending on which page of UDG characters is in the UDG file at the time, the "a" will display and PRINT the first character of that page or a space if that character is undefined in the UDG file.

Line Notes:

Listing 1

9004-9005: Provide an uninterrupted scroll from the bottom line upward.

9015: "bin" is a token word (as on the ZX/TS2068 computers the underlined words in the PRINT statements may be obtained by first entering the token word THEN, followed by the desired token word. The deletion of the THEN is a bit tricky as are line numbers since the deletion of the token word DELETE must be done with the same keystroke.).

9030: INKEY$ input is in binary digits with a restart on a character if "e" (error) is input. Hold the key down for at least 1/60 of a second or INKEY$ will be read as an invalid value.

9040: In a multiple statement line any prior condition set at the beginning of the line applies to the operation of the statements following in the same line.

9070: A temporary attribute may be assigned by making the line read:

IF a$="0" THEN PRINT INK 6; " ";

The INK attribute reverts to original on the next statement.

9710: SAVEs the definition codes separately since the UDG file does not SAVE along with the listing.

9720: The autoLOAD instruction in line 9700 is not activated until after the program is reLOADed back from tape with either LOAD "bin" or LOAD "".

Listing 2

1: REM nn must not be overwritten by any program in which the UDG routines are used.

Listing 3

9230: Allows restarting the program at any time by GO TO menu.

9245: Will appear in the main listing after MERGEing. It must be deleted *before* reSAVEing the main listing back to tape, otherwise the LOADing sequence will be interrupted.

9480: Data is code for the LDIR block transfer routine to and from UDG file.

Listing 3. Page routine.

```
   1 REM ??
9002 LET r=97
9004 POKE 23692,-1
9005 PRINT AT 21,0
9006 PRINT "PAGE NO. ";pno;" INP
UT e FOR ERROR"
9007 PRINT "12345678 INPUT a TO
POKE TO PAGE"
9008 FOR g=r TO 117
9009 FOR i=1 TO 8
9010 LET byte=0
9015 LET b$="BIN "
9020 FOR n=7 TO 0 STEP -1
9030 PAUSE 0
9035 LET a$=INKEY$
9040 IF a$="e" THEN LET r=g: GO
TO 9005
9045 IF a$="a" THEN GO TO 9340
9050 IF a$<>"1" AND a$<>"0" THEN
 GO TO 9030
9055 LET b$=b$+INKEY$
9060 LET byte=byte+2↑n*VAL a$
9070 IF a$="0" THEN INK 6: PRINT
 "█";: INK 0
9080 IF a$="1" THEN PRINT "█";
9090 NEXT n
9100 PRINT i;TAB 10;b$;TAB 24;by
te
9110 POKE USR CHR$ g+i-1,VAL b$
9130 NEXT i
9140 PRINT TAB 5;CHR$ g;"  ";CHR
$ (g+47)
9150 NEXT g
9155 PRINT "PAGE NO. ";pno;" COM
PLETED TO "
9160 LET pno=pno+1
9170 GO TO 9340
9200 SAVE "bin" LINE 9210
9210 LET r$="PEEK 23635+PEEK 236
36*256"
9220 LET nop=PEEK (VAL r$+5)
9230 LET menu=9300
9240 LOAD "bin"CODE 65356-168*no
p,168*nop
9300 PRINT "(MAX. pp = ";nop;")
READ PAGE NO. ";
9310 INPUT pno: PRINT pno
9315 LET v=65356-pno*168
9320 LET phi=INT (v/256)
9330 LET plo=v-phi*256
9335 GO TO 9400
9340 RANDOMIZE USR 65356
9400 RESTORE
9410 FOR k=1 TO 2
9420 FOR n=65356 TO 65367
9430 READ a
9440 POKE n,a
9450 NEXT n
9460 RANDOMIZE USR 65356
9470 NEXT k
9480 DATA 1,168,0,17,88,255,33,p
lo,phi,237,176,201,1,168,0,17,pl
o,phi,33,88,255,237,176,201
9600 PRINT "PAGE ";pno
9610 FOR n=144 TO 164
9620 PRINT CHR$ n;
9630 NEXT n
9640 PRINT '"abcdefghijklmnopqrs
tu"
9650 PRINT '''" INPUT NEW CHR$ A
T (a TO u)"'" INPUT x IF NEW PAG
E"'" INPUT s TO SAVE CODE "
9660 PAUSE 0
9665 LET a$=INKEY$
9670 LET r=CODE a$
9675 IF a$="x" THEN GO TO 9300
9680 IF a$="s" THEN GO TO 9999
9690 GO TO 9004
9999 SAVE "bin"CODE 65356-168*no
p,168*nop
```

TS2068 Programming Tips

Sharon Zardetto Aker

Although the TS2068 users manual does an outstanding job in acquainting the user with the computer, obviously it cannot include everything. Articles in *SYNC* will help you increase your ability to use this new machine. Let's begin by looking at a few tips and techniques to add to your repertoire.

Variable Names

When it comes to variable names, the computer sees no difference between upper and lower case letters, so

```
LET a=5:PRINT A
```

will cause the computer to print a 5.

This holds true even for longer variable names, so SCORE, score, Score, and SCorE are all the same to the computer—not as *strings,* but as variable names.

Attribute Assignment

Four main attributes can be assigned to a character: PAPER and INK colors, and FLASH and BRIGHTness. In addition, OVER and INVERSE can be assigned to any character. PAPER and INK commands are used with color numbers, while the other attributes are used with a 1 to turn them on or a 0 to turn them off.

When an attribute is assigned as a separate command, e.g.,

```
INK 2:PRINT ''Hello''
```

it will remain in effect—it becomes the permanent attribute—until another command is issued to change it. The manual points out that you can use the syntax.

```
PRINT INK 2;''Hello''
```

so that the INK command will affect only that print statement and not change the permanent color in the computer's memory.

Actually, you can combine any of the attribute statements with any of the print or graphic commands.

Sharon Zardetto Aker, 20 Courtland Dr., Sussex, NJ 07461.

SCREEN$ can be used not only to save and load screen displays, but also to check what is appearing in any screen position.

Some of the possible combinations are:

```
PRINT FLASH 1;''Flash''
DRAW INK 5;50,50
CIRCLE BRIGHT 1;127,87,50
PLOT PAPER 4;60,80
```

Remember that an attribute is assigned to a whole character cell—the block of 8 by 8 pixels that a letter occupies. So, there is a vast difference between:

```
CIRCLE 127,87,50
```

and

```
CIRCLE FLASH 1;127,87,50
```

The latter will draw a circle with the usual thin line, but the flash command will affect all the character blocks that the line passes through.

You can use this situation to your advantage in many cases. The following program, for instance, will plot a line the thickness of a character cell, which is more effective for a bar graph than a thin line:

```
10 PAPER 1:CLS:INK 5
15 FOR y=1 TO 30
20 PLOT PAPER 5;10,y
25 NEXT y
```

INPUT LINE

The INPUT LINE variation of INPUT suppresses the quotation marks of the input prompt. If you use

```
INPUT LINE a$
```

only the L cursor will show at the bottom of the screen.

This function has the advantage of preventing the use of STOP at this point of the program. Normally, you can delete the left quotation mark when the computer is waiting for an input and enter STOP; with the quotation marks missing, the STOP will be interpreted as a string rather than as a command. (For some reason, however, pressing shifted 6 while the computer is waiting for the input *will* stop the program.)

When you want to use INPUT LINE with a string, the proper syntax is:

```
INPUT''What is your
name'';LINE a$
```

SCREEN$

SCREEN$ is used to save and load screen displays, but it can also be used to check what is appearing in any position on the screen.

```
PRINT SCREEN$(10,15)
```

will return the character printed in the center of the screen.

The SCREEN$ function is most useful in an IF-THEN construction:

```
IF SCREEN$(5,4)=''A''THEN...
```

SCREEN$ has a few limitations: an inverse character will be read as its true video counterpart, and graphic symbols —from the graphic set or user-defined —will not register at all.

OVER

The OVER command does not simply print one character over the other, or the following commands:

```
10 PRINT ''■''
15 OVER 1
20 PRINT AT 0,0;''!''
```

would result in a black square in the corner of the screen.

Actually, what you get is a white exclamation point on a black square.

When OVER is in effect and two characters are being PRINTed at the same spot, the following rules apply:

1) Where both characters are PAPER color, the combined character remains PAPER color.

COMPUTERIZE YOUR SMALL BUSINESS ACCOUNTING

TIMEX SINCLAIR 2068 COLOR COMPUTER

☐ **ACZ GENERAL LEDGER 2.000**—An advanced, yet easy to use accounting system for your 2068: Prints the Income Statement (P&L), Balance Sheet, Equity Statement, Journal Entries, Ledger Detail and the Chart of Accounts. Supports both monthly and year-to-date statements. Up to 150 accounts and 800 entries and accounts per month. Includes a built-in interface for the optional Cardco numeric keypad (see below).

The comprehensive instruction manual comes in a loose leaf binder, and even shows how to let other programs interact with the General Ledger. **$39.95**

☐ **CARDCO Brand NUMERIC KEYPAD**—Plugs into your joystick port. Includes a software routine that converts the STICK values into numeric input. No hardware modifications needed. (Not for use with cartridges). Lifetime mfg. guarantee. **$44.95**

TIMEX SINCLAIR 1500, 16K TS1000, ZX81

☐ **ACZ GENERAL LEDGER**—This system prepares a full set of financial statements and keeps a record of ledger transactions. Up to 400 entries and accounts per session. $29.95

☐ ACZ CHECK REGISTER—Balances checkbook, reconciles with bank statement, lists checks and deposits in 20 accounts and more. **$12.95**

☐ **ACZ EASY GRAPH**—Simplifies bar graph preparation. Features a special routine that draws a full screen graph in under 10 seconds. **$12.95**

COTTAGE TECHNOLOGY

5720 W. Little York, Suite 178, Houston, Texas 77091

Check items above. Use entire ad as order form.

Add $1.50 shipping and handling on orders under $50.

Amount Enclosed$ ________

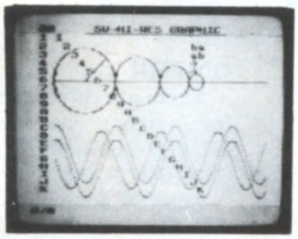
Name ________

Address ________

City/State ________ Zip ________

Texas residents add 5% sales tax. All prices USA.

CIRCLE 15 ON READER SERVICE CARD



Software Only High Resolution Graphics

ZX81 TS1000 TS1500 16-64K

THE SW-HI-RES PACKAGE GIVES YOU CONTROL OVER YOUR SCREEN AS NEVER BEFORE!!!!!!!!!!!

--*-*-*-*-*-*

- Now you can have HI-RES graphics on your TV WITHOUT ANY HARDWARE CHANGES.
- Create your own character/symbol set (Lower Case, bombs, invaders,etc...)
- Access to 174 x 255 pixels.
- Easy to use; access is similar to BASIC PLOT and PRINT.
- Includes 11 users friendly utilities: PLOT,UNPLOT,PRINT,PRINTC,DRAW,CLS, SCROL-UP,-DOWN,-LEFT,-RIGHT,COPY
- Printable on Timex/Sinclair Printer
- In addition to the SW-HI-RES package you will get the BASIC program used to generate the above TV picture.

$20.00

SW Hi-Res 19.95

HI-RES WORD PROCESSOR

*** RATED A+ in SYNC JANUARY/FEBRUARY REVIEW **

—Lower case on the screen (No hardware changes required)

—Print 64 characters per line (Sideways) on T/S Printer (UPPER & lower case)

—Printable on 80-column printer (UPPER & lower case)

—*Fast* word processor with full cursor movement on the screen

HI-RES WP 24.95

SEND CHECK OR M.O. (ADD $1.00 S&H) TO

N. ELMALEH
5100 HIGHBRIDGE STREET 53,
FAYETTEVILLE, N.Y. 13066

CIRCLE 12 ON READER SERVICE CARD

---

50 easy new games, graphics and business programs for your Timex Sinclair 1000

Start getting more fun, more challenge and more practical value from your computer with the most comprehensive book ever written for the TS 1000.

In *50 Programs for the Timex Sinclair 1000*, Leland B. Carter outlines exciting new ways to use your unexpanded Timex for games, graphics, math, physics, business, puzzles and more.

You'll find hours of sheer enjoyment. You'll instantly expand your software library. What's more, you'll permanently enhance your computer programming skills.

Step-by-step instructions in Sinclair Basic let you immediately type in and run programs like Booby Trap...Calendar...2-D Reversal...Calculator...Target Gallery...Blast-Um...Flight Graphic...ESP...Stopwatch...What's the Pattern...and many, many more!

Each program has been thoroughly tested and re-tested for ease of entry. You'll find excellent user-friendly statements, plus better-than-ever graphic and screen display quality. And at just $6.95, you're getting each program for less than 14 cents.

Start getting more from your computer by ordering your copy of *50 Programs for the Timex Sinclair 1000* today!

Compatible for use with Sinclair ZX81

For faster service,
PHONE TOLL FREE
800-631-8112
(In NJ only: 201-540-0445)

MAIL TODAY TO:
CREATIVE COMPUTING PRESS
Dept. NI4H 39 East Hanover Avenue
Morris Plains, NJ 07950

Please send me______ copies of *50 Programs for the Timex Sinclair 1000* for only $6.95 each, plus $1.50 shipping & handling per book. 15T

Total Amount $________.

☐ Payment Enclosed. (CA, NJ and NY State residents please add applicable sales tax.)

Name________ (please print)

Address________

City________ State________ Zip________

*Outside U.S., add $3 for shipping and handling.

☐ Check here to receive a FREE catalog of computing books, magazines and guides.

Also available in your local bookstore or computer store.

2) Where *only one* of the characters is INK color, the combined character is PAPER color.

Try:

```
10 CIRCLE OVER 1;127,87,50
20 GO TO 10
```

This program repeatedly draws a circle in the same spot, but, since OVER is in effect, the circle is erased on every other loop.

You can use PLOT OVER 1 in the same way to erase a plotted line.

DEF FN, FN

With DEF FN and an appropriate variable (a single-letter numeric or string variable), you can define your own mathematic and string functions, calling them up later with FN. (Both DEF FN and FN are accessed by shifting into the extended mode, holding down symbol shift, and pressing 3 or 4.)

If, for instance, you should want to raise a number to the power of five and find the square root of the result, you would use: `PRINT SQR(x↑5)`

If you want to do that same operation on different numbers, you can define a "square root of the fifth power" function:

```
DEF FN a(x)=SQR(x↑5)
```

The variable "a" is the function variable. The parenthetical "x" stands for the number that the function will be performed on. The function is defined by using the parenthetical variable in the mathematical expression.

Now, if you want to apply the function to the number 23, you would use:

```
PRINT FN a(23)
```

You can define a function in terms of more than one variable. For instance, if you want to average three numbers, you can define an averaging function like this:

```
DEF FN a(x,y,z)=(x+y+z)/3
```

Use this function in this way:

```
PRINT FN a(12,17,43)
```

You can also define string functions. If you want (for some strange reason) to form a new string out of the first and last letters of an old string, you can define that function as:

```
DEF FN a$(z$)=z$(1)+z$
(LEN z$)
```

In this case, a$ is the function name, and z$ stands for the string that will be operated on. The slicing operation that is defined takes the first letter of z$ and connects it to the last letter.

```
10 DEF FN a$(z$)=z$(1)+z$
(LEN z$)
15 LET b$=''HELLO''
20 LET c$=FN a$(b$)
25 PRINT c$
```

ON ERR

There are three forms of the ON ERR (on error) command: ON ERR GOTO/GOSUB, ON ERR CONT, and ON ERR RESET.

The ON ERR GO TO and ON ERR GO SUB commands prevent the usual stopping of the program with an error report. The ON ERR statement is placed at or near the beginning of the program—any spot, really, as long as it is *before* the spot where the error might occur.

One of the great advantages of the ON ERR statement is that it can prevent someone from BREAKing into your program, because BREAK is considered by the computer to be an error.

This program draws and erases a circle on the screen repeatedly; you will not be able to use BREAK.

```
10 ON ERR GO TO 15
15 CIRCLE OVER 1;127,87,50
20 GOTO 15
```

Since line 10 instructs the computer to go to 15 when an error is encountered, you will have to shut off the computer to regain control.

You might have a five-item menu on the screen for the user to choose from:

```
10 ON ERR GO TO 20
20 IF INKEY$='''' THEN
GOTO 20
30 IF VAL INKEY$<1 OR VAL
INKEY$>5 THEN GO TO 20
```

Line 30 prevents the computer from accepting any number other than those from 1 to 5; line 10 prevents the user from BREAKing into the program. Unfortunately, line 10 will also prevent *you* from breaking into the program to make any improvements. So, you can add:

```
25 IF INKEY$=''Z''
THEN GOTO 100
```

The chance that someone would enter a capital Z when presented with numerical menu choices are minimal, but you can use it to get to a subroutine that will allow you to STOP the program.

```
100 ON ERR RESET:STOP
```

The RESET command enables the usual errors to be detected and acted upon once again, so the STOP command will be executed. (STOP, like BREAK, is considered an error.)

You can use ON ERR GO TO and ON ERR GO SUB in similar situations, as well as in debugging your programs.

ON ERR CONTINUE causes the program to resume from the line at which the error occurred.

```
5 ON ERR GO TO 25
10 LET a=0
15 PRINT INK a;''hello''
20 LET a=a+1:GO TO 15
25 LET a=0:ON ERR CONTINUE
```

In this program, an error occurs when the ink color is higher than 9 (8 is for transparent, 9 is for contrast.) The error sends the program to line 25, where the variable is reset to zero and the command is given to continue from the point where the error occurred.

A Demonstration

"Random Draw" is a program that uses some of the tips presented in this article. Once you RUN it, you will not be able to stop it, so you might want to save it before RUNning.

The program will draw randomly colored lines of random lengths on a black background. The first line starts at the center of the screen, going off in a random direction; each subsequent line begins where the last ended. ON ERR is used to send the program back to the center of the screen if a line has gone off an edge. Let the program run a while and you will notice that instead of pixel-thin lines being drawn, blocks begin to fill the screen and change color. This is an example of how attribute assignment affects an entire character cell, and not just the pixels you are drawing or plotting. As a new line passes through a character cell, all the other lines in that cell are changed to the current color.

```
1 REM random draw
5 PAPER 0: CLS : BORDER 0
10 DEF FN r(x)=INT (RND*x)
15 ON ERR GO TO 20
20 PLOT 127,87
25 LET a=FN r(150)-75
30 LET b=FN r(150)-75
35 LET i=FN r(7)+1
40 DRAW INK i;a,b
45 GO TO 25
```

Line Notes

5: Sets the entire background to black.

10: Defines a "random" function so that the random number will be one of x different integers (beginning at a default of 0).

15: When an error occurs—when a line is going to be drawn off-screen—the program is sent back to line 20, which lets the drawing start at the center of the screen again.

25,30: Define a and b as random numbers between -75 and +75 (150 different numbers, beginning at -75)

35: Defines i as a random number between 1 and 7.

The Circle Game

Harry Doakes

We will tackle a relatively complicated challenge: getting the TS1000 to draw a circle on the screen—both the hard way and the easy way.

Of all the problems with machine code programming, there is one that really stands out: it is too *simple!*

Oh, sometimes it can be pretty hard to figure out how to make your program do what you want. But that is because the Z80 microprocessor—the "brain" inside your ZX/TS computer—is really rather simple-minded. It can add and subtract well enough—and even multiply and divide by two—as long as the numbers are between 0 and 65,535.

But for anything more complicated, the Z80 just cannot handle it alone. You—the programmer—have to do most of the work. You are the one who has to write a routine that will multiply or divide, or print something on the screen. If you want the Z80 to do something *really* fancy like square roots or trigonometric functions or floating-point math, you have to figure out some way to simplify it all down to very simple-minded jobs like adding and subtracting.

Either that, or you have to get someone else to do it for you.

In this and the next installment of this series of articles on Z80 machine code, we will try it both ways as we tackle a relatively complicated challenge: getting the ZX81 or TS1000 to draw a circle on the screen. This time we will do it the hard way, translating a circle drawing routine into machine code ourselves. Next time, we will let someone else do most of the work—using a special section of the Basic ROM called the "calculator."

Harry Doakes, PO Box 10860, Chicago, IL 60610.

Why do it yourself if you can use ROM routines to do it more easily? Because the ROM "calculator" is only about twice as fast as Basic would be. "Pure" machine code is *much* faster than that. There is another reason, too: sometimes you may want to do something that the "calculator" in ROM just cannot handle. With regular machine code, you can do almost anything—if you can only figure out *how*...

Round and Round

Figure 1 shows one of the most common ways of drawing a circle using Basic. Type in the program and RUN it. You will see it slowly draw a circle, PLOTting one point at a time, on your screen.

Figure 1.
Common circle drawing routine in Basic.

```
10 LET R=20
20 LET S=32
30 LET T=22
40 FOR N=0 TO 2*PI STEP PI/60
50 LET X=R*COS N
60 LET Y=R*SIN N
70 PLOT X+S,Y+T
80 NEXT N
```

Figure 2. Alternate circle drawing routine.

```
10 LET R=20
20 LET S=32
30 LET T=22
40 LET X=R
50 LET Y=0
60 FOR N=1 TO 202
70 LET X=X-Y/32
80 LET Y=Y+X/32
90 PLOT X+S,Y+T
100 NEXT N
```

This circle drawing routine is pretty simple. It is based on a couple facts from trigonometry that show up in lines 50 and 60:

X=R*COS N
Y=R*SIN N

Or, as your trigonometry teacher might put it, "For any angle, the x-coordinate of a point on the circle equals the radius times the cosine of the angle, and the y-coordinate equals the radius times the sine of the angle." If you have never had a trigonometry teacher, do not panic. SIN and COS are too complicated to explain here in detail, but, even if you do not understand exactly how they work, the program will run anyway.

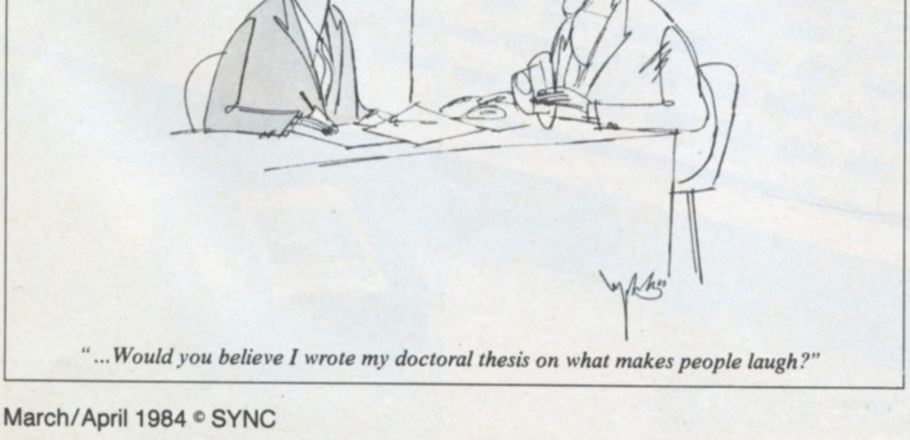
In this routine, the radius of the circle is variable R. The LET command in line 10 sets it to 20. The center is at point (S,T); lines 20 and 30 make that (32,22), the center of the screen. You can change lines 10, 20, and 30 to make larger or smaller circles, with centers in different places on the screen. Watch out, though; if your routine tries to plot a point that is off the screen, you will get an error, and the program will stop.

There are really only two problems with this routine: SIN and COS. If the sine and cosine functions are complicated to explain, they are even *more* complicated for the computer to work out. The routine draws a circle so slowly because it spends most of its time figuring the sines and cosines.

Fortunately, we are not stuck with using these functions for drawing a circle. There are other ways to do it. Probably the easiest is the one worked out by Marvin Minsky, a computer scientist at the Massachusetts Institute of Technology. (You may have heard of Minsky; he is most famous for his work on an "artificial intelligence" computer language called LISP.)

Figure 3. Translation of Figure 2 into MC.

Basic program		MC instructions
10 LET R=20		(you can load these
20 LET S=32		three variables with
30 LET T=32		their values before
		the routine begins)
40 LET X=R		LD A,(R)
		LD H,A
		LD L,0
		LD (X),HL
50 LET Y=0		LD H,0
		LD (Y),HL
60 FOR N=1 TO 202		LD A,1
		LD (N),A
70 LET X=X-Y/32		LD HL,(X)
		LD DE,(Y)
		LD B,5
	DIV1:	SRA D
		RR E
		DJNZ DIV1
		OR A
		SBC HL,DE
		LD (X),HL
80 LET Y=Y+X/32		LD DE,(Y)
		LD B,5
	DIV2:	SRA H
		RR L
		DJNZ DIV2
		ADD HL,DE
		LD (Y),HL
90 PLOT X+S,Y+T		LD DE,(X)
		LD A,(S)
		RL E
		ADC A,D
		LD C,A
		LD A,(T)
		RL L
		ADC A,H
		LD B,A
		CALL PLOT
100 NEXT N		LD A,(N)
		INC A
		LD (N),A
		CP 202
		JR C,line 70



"...Would you believe I wrote my doctoral thesis on what makes people laugh?"

A few years back, Minsky was looking for a better way to draw a circle with a computer—a way that would be simpler for the computer to do. He came up with a routine that, in Basic, looks something like Figure 2.

It certainly *seems* like a much simpler routine, that's for sure. There is no more SIN or COS—just adding and subtracting, multiplying and dividing. Since simpler means faster on a computer, it can draw a circle in much less time than the SIN and COS version. Type it in and run it, and you will see just how much faster it is.

You may wonder how it can do the work of SIN and COS when it is so simple. Actually, it is just looks simple. The ideas behind it are quite complex—even more complicated than the ones behind using the sine and cosine functions. But this is not a course in college-level math; if it were, this issue of *SYNC* would be 500 pages long. Just remember that, like SIN and COS, it *does* work—even if you do not understand exactly how.

A Circle in Any Language

Translating this circle program into Z80 machine code really is not too difficult. It is *long,* that is true—but, if you translate the Basic program one line at a time, it is not hard. The translation is in Figure 3.

Let's look at how some of the lines are translated; they are not all exactly what you might expect. Line 70, e.g., seems to be a pretty complicated piece of translation:

```
70 LET X=X-Y/32
```

Subtracting is easy, but how do we divide by 32?

We already know how to divide a number by 2. We can use the Z80 is "shift" and "rotate" instructions. A shift instruction, by itself, will divide the number in a register by 2. Using shift and rotate instructions together, we can do the same thing to the number in a register pair, such as DE or HL. For example, the instructions

```
SRA D
RR E
```

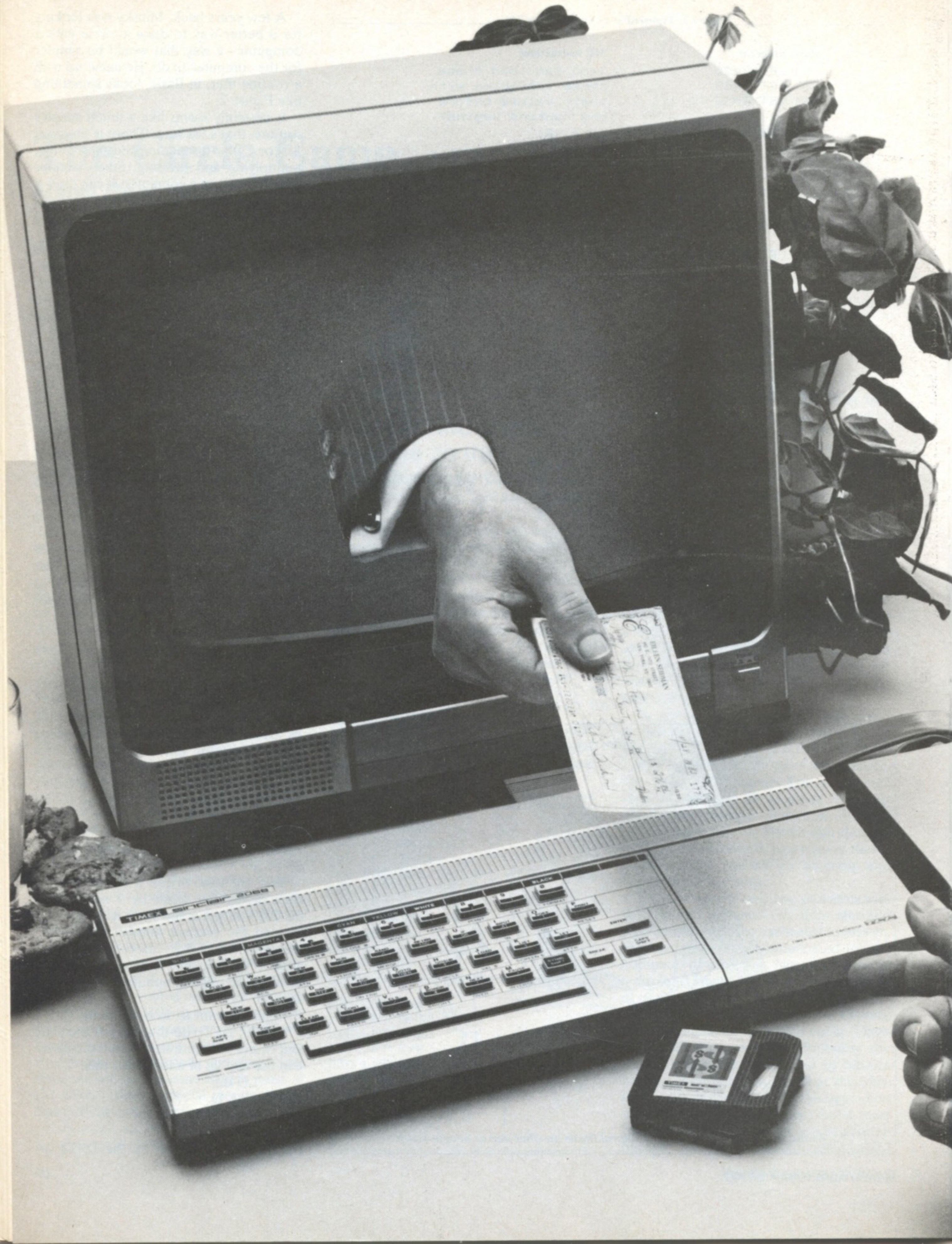
divide the number in register pair DE by 2.

Now try this: divide a number by 2, then divide the result by 2 again. The answer you get is the same as if you had divided the original number by 4. (Remember, 4 = 2 * 2.) If you divide that answer by 2 again, it is as if you had divided the original number by 8 (2 * 2 * 2).

What we want to do is divide by 32 (2 * 2 * 2 * 2 * 2). To do this, we put the shift and rotate instructions that will divide by 2 in the middle of a loop, like this:

```
      LD B,5
DIV1: SRA D
      RR E
      DJNZ DIV1
```

Do not be thrown off by the DJNZ in-


TIMEX sinclair 2068

struction. It is a special instruction that only works with register B; it is exactly like the two instructions

```
DEC B
JR NZ,DIV1
```

Every time the Z80 gets to the DJNZ instruction, it will subtract one from the number in register B. If the result is greater than zero, the program will jump; if it is zero, it just continues with the next instruction.

In other words, since register B starts out with the number 5 we will run through this loop 5 times. Each time the number in register pair DE will be divided by 2, and the number in register B will drop by 1, until finally B contains zero—and the program continues. In Basic, it might look something like this:

```
10 LET B=5
20 LET DE=DE/2
30 LET B=B-1
40 IF B>0 THEN GOTO 20
```

As you have probably noticed, the DJNZ instruction works a lot like the NEXT command in Basic. Here is another Basic version of the routine:

```
10 FOR B=5 TO 1 STEP-1
20 LET DE=DE/2
30 NEXT B
```

If you add a couple more lines such as

```
5 INPUT DE
50 PRINT DE
```

to either of these short programs, you can try running them in Basic. You will find that, as expected, each of them divides by 32.

The Fixer

There is something else that is a bit unusual in this translation. Though the variables R, S, and T are ordinary numbers, X and Y use a very special kind of arithmetic called *fixed point* arithmetic.

Here is why: when the plotting section of this routine starts its work, X equals 20 and Y equals 0. The first thing we do is divide Y by 32 and subtract the result from X:

X=X-Y/32
X=20-0/32
X=20

So far, so good. Next we will divide X by 32 and add the result to Y:

Y=Y+X/32
Y=0+20/32

Here is where the problem shows up. The Z80 microprocessor only uses *integers*, i.e., whole numbers. There are not any fractions in this kind of arithmetic. When you divide a number, the result is always rounded off to the next whole number down. It is as if you have used the INT function in Basic.

This means that, for the Z80, 20/32 equals zero. And *this* means, after all that work, X still equals 20, and Y still equals 0. No matter how many times we run these numbers through the loop, they will never change. That is not going to draw much of a circle.

How can we solve this problem? We will cheat. We can trick the computer into keeping at least part of a fraction by putting each of these two variables into the *top* half of a register pair. In the translation of our Basic program, we do that in lines 40 and 50.

Now remember: putting a number in the top half of a register pair is like multiplying it by 256. So *to the computer,* X is not 20 any more; it is 20 times 256, or 5120. (Y, of course, is still zero since 0 times 256 is still 0.)

When we run these two numbers through our routine now, the fractions do not completely disappear. They are still there in the *lower* half of each register pair:

X=X-Y/32	Y=Y+X/32
X=5120-0/32	Y=0+5120/32
X=5120	Y=160

This time, we do not end up with exactly what we started with. The second time through, the numbers change even more

X=X-Y/32	Y=Y+X/32
X=5120-160/32	Y=160+5115/32
X=5115	Y=319

—and we are off and running.

"Fixed point" arithmetic is not as accurate as "floating point," and it works with only a limited range of numbers.

To PLOT these points, of course, we have to use the actual values of X and Y. To find them, we must divide by 256, but that is easy to do. We just use the value that is in the top half of each register pair.

In Basic, what we are doing would look something like this:

```
LET X=X*256
LET Y=Y*256
FOR N=1 TO 202
LET X=X-Y/32
LET Y=Y+X/32
PLOT (X/256)+S,(Y/256)+T
NEXT N
```

Get the idea? The computer thinks it is working with whole numbers, but we know that they are really fractions. This kind of arithmetic is called "fixed point" because it is as if we had stuck a decimal point between the top and bottom halves of the register pair. The imaginary decimal point is always in the same place; this is why we say it is "fixed."

This fixed-point arithmetic is not as accurate as the floating point arithmetic that Basic uses, and it can only work with numbers in a limited range; in this case, e.g., X and Y cannot be less than -127 or more than 127. But fixed point arithmetic is an easy way to use fractions in machine code, and it is much faster than floating point.

There is something else you should notice about using these fixed point versions of X and Y—something that shows up when we translate PLOT from Basic to machine code. When we use the PLOT function in Basic, it ordinarily rounds the horizontal and vertical coordinates to the closest whole number. (Remember, PLOT cannot use fractions, either.)

For example, the Basic command

```
PLOT 16.8,7.2
```

is actually treated as if it were

```
PLOT 17,7
```

Since 16.8 is closer to 17 than 16, it is rounded up. 7.2 is closer to 7 than to 8, so it is rounded down instead.

In the machine code version, we *could* use instructions like these:

```
LD DE,(X)
LD A,(S)
ADD A,D
```

But with these instructions, only the whole number part of X would be added to S. Instead of being rounded to the *closest* number, X would always be rounded *down*. The result would look like the circle we get when line 90 of the Basic version is

```
90 PLOT INT (X+S),INT (Y+T)
```

Try running the Basic program again with this new line. You will see that rounding down makes a circle that is, well, not as round.

To smooth things out, we have added another step. We replace

```
ADD A,D
```

with

```
RL E
ADC A,D
```

These instructions round the number X+S to the nearest integer, like PLOT does. It is only a little more work, and it makes a much better looking circle.

Circling the Wagons

A routine that draws a circle is nice, but it is not really a circle drawing function. As it stands, this machine code subroutine does not offer any way to make circles with a radius other than 20, or with any other center than (32,22).

Fortunately, as with the Basic version, it is easy to change the machine code variables R, S, and T by POKEing numbers into those variables from Basic before our program hits the USR function.

But that takes time, and it is complicated and inconvenient, too. A better way would be to let the machine code routine get its own information from the Basic program. If we could put the information in the same line of the Basic program as the USR function that starts the circle drawing routine, it would be easier and simpler to use—and faster, too.

One way to do that is to make both the USR function and the information part of an IF-THEN statement. It might look something like this:

60 IF USR (Q) THEN REM R5 S13 T7

As you might guess, "R5" would set the radius variable R to 5; "S13" and "T7" would set S to 13 and T to 7, putting the center of the circle at (13,7) on the screen.

Here is how we can use a line like this, first from point of view in Basic, then from the machine code side.

When Basic gets to IF, it expects the next thing in the line to be an expression like "A=5", a statement that will either be true or false. (If a number follows IF instead of an expression, it is still either true or false to the computer, i.e., it is "false" if the number is zero, and "true" otherwise.) If it is true—or "true"—then Basic performs the command that follows THEN; otherwise, that part of the line is skipped. In this case, if does not matter whether Basic thinks USR (Q) is true or false. The THEN part of the line is just a REM statement. Either way, Basic will skip over it.

But the USR function is in the IF part of the line, so Basic evaluates it anyway, i.e., the computer performs the machine code routine. Then it goes on to the next line of the program.

In other words, we can safely put our circle information in that REM statement, and it is even on the same program line as USR (Q). But how will the machine code routine know where to look for that information?

We are in luck for that: the system variable "CH-ADD" always keeps track of where the computer is in a Basic program. This variable is stored at locations 16406 and 16407 on the TS1000, ZX81, or ZX80 with 8K ROM and at 16422 and 16423 on the ZX80 with 4K ROM.

This is a two-byte pointer, so it can hold any value from 0 to 65535. This means it can point to any location in memory. In fact, it is always pointing to the section of your Basic program that is about to be performed, e.g., whenever Basic gets to a USR function, CH-ADD points at whatever immediately follows it. So, if a line starts out "IF USR (Q)", CH-ADD points to what is just after the USR function, i.e., at the THEN statement. Now remember: THEN, like all the other keywords in Timex and Sinclair Basic, takes up only one byte of memory. So does REM.

Suppose we write a machine code routine that copies the number that is in CH-ADD; that number points at THEN. If we add 1 to this number, it will point at REM. Increment it again, and it will point at the first character in the REM statement which is where we have put the information we want the machine code routine to find.

In other words, your machine code routine can easily find where the information is in memory; all it takes is three Z80 instructions:

```
LD BC,(CH-ADD)
INC BC
INC BC
```

Reading Lesson

Now that you know where the information is, it is relatively easy to get it out of the REM statement. Figure 4 is a Basic routine that demonstrates how it is done.

The routine is a bit long, but it is straightforward. Lines 30-90 figure out what variable the information belongs to; then lines 100-180 convert the characters in the REM line to a number, which is PRINTed by line 190. Run this program, then try it with different values for R, S, and T.

Figure 5 is the machine code translation of the routine. This time, instead of being PRINTed, the numbers are loaded into the appropriate machine code variables. You should be able to use it as a model, so your own routines can get information from the REM statement in an IF-THEN line.

This routine will work with either the 8K ROM or the 4K ROM (as long as you remember that each ROM keeps the CH-ADD system variable in a different location).

The 4K ROM does not have the PLOT function, though, so it cannot use the circle drawing routine. There is something else the 8K ROM allows you to do that the 4K ROM will not: with the TS1000 or ZX81, you do not have to put your machine code routine at the top of memory. Instead, you can POKE a machine code routine into a REM statement at the beginning of your program. That means you do not have to go to the trouble of reserving space as we have done in the past; your machine code routine will be protected as part of the Basic program.

Furthermore, we do not need a different version of the routine for different amounts of RAM; as long as it is in the first line of your program, the machine code will always start in the same place.

Figure 4.
Basic program to get information out of REM statement.

```
 10 REM R5 S13 T7
 20 LET BC=16514
 30 LET BC=BC+1
 40 LET A=PEEK (BC)
 50 IF A=CODE("R") THEN LET HL=
CODE("R")
 60 IF A=CODE("S") THEN LET HL=
CODE("S")
 70 IF A=CODE("T") THEN LET HL=
CODE("T")
 80 IF A=0 THEN GO TO 30
 90 IF A=118 THEN GO TO 210
100 LET D=0
110 LET BC=BC+1
120 LET A=PEEK (BC)
130 IF A<CODE("0") THEN GO TO 1
90
140 IF A>CODE("9") THEN GO TO 1
90
150 LET A=A-28
160 LET A=A+D*10
170 LET D=A
180 GO TO 110
190 PRINT CHR$(HL),D
200 GO TO 40
210 STOP
```

Figure 5.
MC translation of Figure 4.

```
Basic program                  MC instructions

 10 LET BC=PEEK (CH-ADD)       LD BC,(CH-ADD)
 20 LET BC=BC+1                INC BC
 30 LET BC=BC+1                INC BC
 40 LET A=PEEK (BC)            LD A,(BC)
 50 IF A=CODE("R")             CP 55
    THEN LET HL=location       JR NZ,line 60
    of variable R              LD HL,R
 60 IF A=CODE("S")             CP 56
    THEN LET HL=location       JR NZ,line 70
    of variable S              LD HL,S
 70 IF A=CODE("T")             CP 57
    THEN LET HL=location       JR NZ,line 80
    of variable T              LD HL,T
 80 IF A=0                     CP 0
    THEN GO TO 30              JR Z,line 30
 90 IF A=118                   CP 118
    THEN GO TO 210             JR Z,line 210
100 LET D=0                    LD D,0
110 LET BC=BC+1                INC BC
120 LET A=PEEK (BC)            LD A,(BC)
130 IF A<CODE("0")             CP 28
    THEN GO TO 190             JR C,line 190
140 IF A>=CODE("A")            CP 38
    THEN GO TO 190             JR NC,line 190
150 LET A=A-28                 SUB A,28
160 LET A=A+D*10               LD E,A
                               LD A,D
                               ADD A,A
                               ADD A,A
                               ADD A,D
                               ADD A,A
                               ADD A,E
170 LET D=A                    LD D,A
180 GO TO 110                  JR line 110
190 POKE HL,D                  LD (HL),D
200 GO TO 40                   JR line 40
210 (next routine)
```

It still takes about 5 seconds to draw the circle, the computer spends quite a while in the ROM subroutine PLOTing the circle's 202 points.

Figure 6 is the full circle drawing program, including the routine that checks for information in an IF/THEN statement. It is designed to start at location 16514—the first character in a REM statement that is the first line of a Basic program.

Use the program in Figure 7 to load the machine code routine into memory. Test it by typing

IF USR 16514 THEN REM R5 S58 T38

It should draw a small circle in the upper right corner of your screen. Once you are sure it works, you can delete the loader program except for line 1. Now you can SAVE the routine on tape or make it part of another program.

It still takes about 5 seconds to draw the circle, even using machine code; the computer spends quite a while in the ROM subroutine, PLOTing each of the circle's 202 points. But it is quite a bit faster than Basic, and that is what machine code is all about.

Figure 6. Full circle drawing program.

```
Machine code              MC instructions
ED 4B 16 40               LD BC,(CH-ADD)
03                        INC BC
03                 LOOP:  INC BC
0A               LOOP+1:  LD A,(BC)
FE 37                     CP 55
20 03                     JR NZ,3
21 1A 41                  LD HL,R
FE 38                     CP 56
20 03                     JR NZ,3
21 1B 41                  LD HL,S
FE 39                     CP 57
20 03                     JR NZ,3
21 1C 41                  LD HL,T
FE 00                     CP 0
28 E5                     JR Z,LOOP
FE 76                     CP 118
28 1B                     JR Z,CIRCLE
16 00                     LD D,0
03                DIGIT:  INC BC
0A                        LD A,(BC)
FE 1C                     CP 28
38 10                     JR C,STORE
FE 26                     CP 38
30 0C                     JR NC,STORE
D6 1C                     SUB A,28
5F                        LD E,A
7A                        LD A,D
87                        ADD A,A
87                        ADD A,A
82                        ADD A,D
87                        ADD A,A
83                        ADD A,E
57                        LD D,A
18 EA                     JR DIGIT
72                        STORE: LD (HL),D
18 C7                     JR LOOP+1
3A 1A 41         CIRCLE:  LD A,(R)
67                        LD H,A
2E 00                     LD L,0
22 1D 41                  LD (X),HL
26 00                     LD H,0
22 1F 41                  LD (Y),HL
3E 01                     LD A,1
32 21 41                  LD (N),A
2A 1D 41            EXT:  LD HL,(X)
ED 5B 1F 41               LD DE,(Y)
06 05                     LD B,5
CB 2A              DIV1:  SRA D
CB 1B                     RR E
10 FA                     DJNZ DIV1
B7                        O A
ED 52                     SBC HL,DE
22 1D 41                  LD (X),HL
ED 5B 1F 41               LD DE,(Y)
06 05                     LD B,5
CB 2C              DIV2:  SRA H
CB 1D                     RR L
10 FA                     DJNZ DIV2
19                        ADD HL,DE
22 1F 41                  LD (Y),HL
ED 5B 1D 41               LD DE,(X)
3A 1B 41                  LD A,(S)
CB 13                     RL E
8A                        ADC A,D
4F                        LD C,A
3A 1C 41                  LD A,(T)
CB 15                     RL L
8C                        ADC A,H
47                        LD B,A
CD B2 0B                  CALL PLOT
3A 21 41                  LD A,(N)
3C                        INC A
32 21 41                  LD (N),A
FE CA                     CP 202
38 BB                     JR C,NEXT
C9                        RET

14                 411A:  R=20
20                 411B:  S=32
16                 411C:  T=22
00 00              411D:  X=0,0
00 00              411F:  Y=0,0
00                 4121:  N=0
```

Figure 7. MC loader routine.

```
   1 REM ED4B164003030AFE3720032
11A41FE382003211B41FE392003211C4
1FE0028E5FE76281B1600030AFE1C381
0FE26300CD61C5F7A87878287835718E
A7218C73A1A41672E00221D412600221
F413E013221412A1D41ED5B1F410605C
B2ACB1B10FAB7ED52221D41ED5B1F410
605CB2CCB1D10FA19221F41ED5B1D413
A1B41CB138A4F3A1C41CB158C47CDB20
B3A21413C322141FECA38BBC91420160
000000000

 10 LET S=16514
 20 LET A=0
 30 LET H=PEEK (S+2*A)-28
 40 IF H<0 OR H>15 THEN STOP
 50 LET L=PEEK (S+2*A+1)-28
 60 IF L<0 OR L>15 THEN STOP
 70 LET N=16*H+L
 80 POKE S+A,N
 90 LET A=A+1
100 GOTO 40
```

Letters, We Get Letters . . .

Several readers have asked, "Is there any difference between PRINT USR B and LET A = USR B and RUN USR B and RAND USR B? When should each one be used?"

To Basic, USR is just another function, like INT or SQR or SIN. It takes one number (the "argument"), does something to it, and results in another number. For example, SQR 4 takes the number 4, finds its square root, and results in the value 2. You can use SQR 4 anywhere you can use the number 2, e.g.,

LET A=SQR 4

will give A the value of 2.

USR works the same way. It is always looking for an argument, and it always produces a number. But, when the computer gets to a USR function in a Basic program, it goes flying off to perform your Z80 machine code routine. When it finally RETurns to Basic, the number that was in register pair BC is sent back to Basic as the result.

In other words, you can use USR B anywhere that you can use a number or a variable in Basic. PRINT USR B will print whatever was in BC on the screen. RUN USR B will treat what was in BC as a line number and try to run the Basic program starting there. RAND USR B will take BC and make it the new "seed" for the random number generator—a pretty harmless thing to do with it.

LET A=USR B

will make A equal to whatever was in BC. In fact, you can use USR B with any command or function that can accept a number; there is a complete list of them in chapter 21 of the Timex manual.

The rule of thumb is this: do not tell the computer to do anything you do not want it to do. If you do not want something on the screen, do not PRINT USR B. If you do not want to lose the number in A, do not LET A = USR B.

Get the idea? Treat USR B like a number, and apply a large helping of common sense. It should not give you any trouble.

Coming Attractions

Next time we will look at the "calculator" in your TS1000 or ZX81 ROM. It is surprisingly easy to use, and with a little practice you can make floating point numbers and complicated math functions part of your machine code programs. We will also take a quick look at machine code programming on the new Timex Sinclair 2068 Color Computer.

If you have comments or questions about machine code programming, or if something is not quite clear, let me hear from you. Be sure to send along a stamped, self-addressed envelope if you need a reply.

In and Out of SYNC

Continued from page 15.

On the TS2068 to control the color that characters are printed in, type in INK and the color number. To illustrate this, change line 20 of the TS2068 program above to:

20 INK 1:PAPER 5:PRINT "In and out of Sync"

This gives blue letters on a cyan background. Remember, you can change any of these in immediate mode.

The following few lines better illustrate the use of INK.

TS2068:

```
100 INK INT (8*RND)
110 PRINT "abcdefabcdef"
120 GO TO 100
```

In line 110, the second half of the string is inverse. Since it is inverse, instead of the characters being the INK color, the PAPER will be the INK color and the character will be the original PAPER color. The INK and PAPER colors are easily reversed by the INVERSE command.

Other commands of the TS2068 that control the display are FLASH, which flashes the specified text, and BRIGHT, which brightens anything printed. Each of these functions is turned on by typing the function followed by a 1, and turned off by typing a 0; e.g.,

FLASH 1:PRINT "HELLO":FLASH 0

Although the TS1000 has is limited to gray, black, and white, still interesting screen displays can be created. The following lines simply fill up the screen with graphics A's. The other (very close) shade of gray is the graphics on the H key. For a black background, use an inverse space.

TS1000:

```
10 FOR I=1 TO 22
20 PRINT "▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
30 NEXT I
```

We have seen that using colors on the TS2068 and Commodore C-64 is easy. The TS2068 uses the easier functions BORDER, PAPER, and INK. Changing text color on the TS2068 does not require POKEing. Using color and changing color on the C-64 is rather cumbersome. The color of the border and background must be changed by POKEing the correct numbers, while changing text colors uses control characters or POKEing into the color memory. The TS1000 can use black, white, and two shades of gray to add at least some flavor to a screen display.

Using RAM Packs

Edward E. Beeler

TS1500 Plus 16K RAM

Getting the TS1500 and the TS1016 RAM pack to work properly involves a little more than plugging everything together and turning the power on. The following steps are suggested:

1) Plug the RAM pack in, type in:

```
POKE 16388,0
POKE 16389,192
NEW
```

These will clear everything (program lines and variables) out of the system and activate the additional 16K from addresses 32768 to 49151. This step has to be performed each time you power up.

2) Verify the existence of the additional RAM to make sure that everything is as it should be, e.g., connectors connected and a working RAM pack. First, enter

```
PRINT((PEEK 16388+256*PEEK
16389 )-16384)/1024
```

This should return a value of 32, representing 32K. It only means that 16388 and 16389 are POKEd correctly, not necessarily that you have 32K of RAM available.

To confirm the existence of our 32K of RAM, type in Listing 1 which will test all addresses from 16K-32K and print out any addresses that fail to pass the test. If you find any errors, unplug everything and start from the beginning. If the errors persist, you can assume that there is a bug in the system. If, on the other hand, everything went smoothly, you are ready to use your machine with confidence.

To better signal that the test is finished, you can add this line

```
85 PRINT AT 10,15;"END TEST"
```

Before we get all excited about all this extra RAM, we need to keep in mind something of how the computer works.

1) No memory exists from 8K-16K (8193-16383). That is right—nothing.

Edward E. Beeler, 9316 Cherry Hill Rd., #1, College Park, MD 20740.

A very long program (around 16K) pushes the display file up to 32767, and the computer thinks it is out of memory.

2) When you write or load a program, the display file is pushed up into memory the length of the listing. This means that, if you enter a program line 10 addresses long, the display file would be moved 10 addresses up from its present address. This is done automatically as the line is entered.

The danger in this is that, if you are entering a very long program (around 16K), the display file is pushed to 32767. When this happens, the builtin memory tester will tell the computer it is out of memory, and the computer will not accept any more input of any kind. In some cases the system completely freezes so that the only option is to unplug and start over.

To keep from getting caught in this predicament, we need to keep track of the VARS file which is right behind the end of the display file. The trick here is to keep the end of the display file away from 32767. One way to do this is to enter as a program line:

```
PRINT 32767-PEEK 16400-256*PEEK
16401
```

Addresses 16400-16401 keep track of the beginning addresses of the VARS file. The above statement will let you know how many bytes are available for programming so just GOTO it from time to time. Keep in mind, 16K for programming is a lot.

Listing 1.

```
 10 FAST
 20 FOR J=32768 TO 49151
 30 POKE J,255
 40 LET A=PEEK J
 50 POKE J,0
 60 LET B=PEEK J
 70 IF A<>255 OR B<>0 THEN GOSU
B 100
 80 NEXT J
 85 PRINT AT 10,15;"END"
100 PRINT "ERROR AT ";J
110 RETURN
```

Using the 16K to 32K Area

The area 16K to 32K (32678-49151) is now available for storing variables, USR routines, any stacks that are pushed up, and data storage. These suggestions will help you to take advantage of the space.

1) Variables and Data Storage

To store variables and data (string variables), you do not need to do much more than describe your intentions to the computer, perhaps in a program line. The machine does the rest.

To get some idea of how this works, type in Listing 2. Press CLEAR and ENTER. Type in GOTO 9910 and ENTER. Record the numbers printed on the screen. Now press RUN and ENTER. Notice the number change of E-LINE. This number changed because E-LINE follows the VARS file and the computer has automatically expanded the VARS file to accommodate the A$ just entered.

Listing 2.

```
 100 DIM A$(5000)
 110 PRINT LEN A$;"  = LEN A$"
9910 PRINT PEEK 16396+256*PEEK 1
6397;" = D-FILE"
9920 PRINT PEEK 16400+256*PEEK 1
6401;" = VARS"
9930 PRINT PEEK 16404+256*PEEK 1
6405;" = E-LINE"
9940 PRINT PEEK 16410+256*PEEK 1
6411;" = STKBOT"
9950 PRINT PEEK 16412+256*PEEK 1
6413;" = STKEND"
9960 PRINT PEEK 16388+256*PEEK 1
6389;" = RAMTOP"
```

Figure 1 shows a typical screen printout. Notice that the difference between the two E-LINE values is a little more than 5000. This is because of the way the computer sets up each variable in order to find it when you so instruct it.

Review Chapter 26 of the TS1000 manual or Appendix D of the TS1500 manual for byte allocations.

Now EDIT line 9960, delete the 0, and create line 9961. ENTER it, and RUN the program again. This time notice the difference in the values of the various files. Everything has been shifted to make room for the new program line. Repeat this process and notice the numbers change again. Remember that D-FILE marks the end of the program area which always starts at 16509.

2) USR Routines (Machine Code)

Some magazine articles suggest storing machine code above RAMTOP. If we are going to do this, we must now lower RAMTOP since previously we opened all the addresses for Basic. (The process for lowering RAMTOP is given in your manual.)

To see how changing RAMTOP works, let's rePOKE 16388 and 16389 which should now contain 0 and 192 respectively. This combination equals 49152. Each decrement in 16389 is worth 256. So

POKE 16389,191

and RUN Listing 2. Notice that RAMTOP is now 48896 which is 256 less. Now let's

POKE 16388,1

and RUN the program again. Notice that RAMTOP is increased by 1. Each value in 16388 = 1 x that value, and each value in 16389 = 256 x that value.

After RAMTOP has been set, all values in addresses above RAMTOP are safe from being accidentally overwritten by something in the VARS file.

We access machine code routines by the USR function which is similar to GOSUB in Basic except that GOSUB tells the computer to go to a specific line in the program while USR tells the computer to go to a specific address in memory, run through the addresses in order until a machine code return statement is encountered, and then continue executing the Basic program.

If you have wondered about those machine code routines stored in REM statement and why they are always accessed by

RAND USR 16514

enter Listing 3 and RUN it. Note in the results (Figure 2) that 16509-16510 indicate the line number; 16511-16512, the number of bytes in the line; 16513, the REM; and 16524 (118), end of line. But 16514 to 16523 can be rePOKEd with a machine code routine. 16514 will always be the usable address after a REM statement when it is the first line of the program. (Enter CONT to see more of your program.)

Figure 1.

```
16864 = D-FILE
17657 = VARS
17658 = E-LINE
17658 = STKBOT
17914 = STKEND
65535 = RAMTOP

5000  = LEN A$
16864 = D-FILE
17657 = VARS
22664 = E-LINE
22664 = STKBOT
22664 = STKEND
65535 = RAMTOP
```

Figure 2.

```
16509  0
16510  1    ▪
16511  12   £
16512  0
16513  234  REM
16514  29   1
16515  30   2
16516  31   3
16517  32   4
16518  33   5
16519  34   6
16520  35   7
16521  36   8
16522  37   9
16523  28   0
```

Listing 3.

```
  1 REM 1234567890
 10 FOR J=16509 TO 60000
 15 PRINT J;"  ";PEEK J;TAB 12;
CHR$ PEEK J
 20 NEXT J
```

The hazard of a REM statement is that you can wipe out the MC routine if you accidentally enter the line number. On the good side, you can SAVE the routine whereas you do not SAVE anything that is held above RAMTOP.

It is time to check the address shift again. Type in GOTO 100 and notice the differences.

TS1000/ZX81 Plus 64K RAM

The procedure for setting up a TS1000/ZX81 with a 64K RAM pack is similar to setting up the TS1500 with the 16K RAM pack. The steps suggested below are for the Memopak 64.

1) Before you connect the RAM pack, you must check the switch settings on the back. One switch must be on at all times, and no more than two can be on at one time. The most common switch combinations is: 2 and 3 on, 1 and 4 off. This yields the highest density of RAM. The other combinations are for different exposures of the 8K-16K area, and one is specifically for the ZX80.

2) With the computer off, plug the pack into the computer and turn it on. Note that it takes a little time for the K cursor to appear. This is because, without doing anything else, the RAM pack has just cleared and activated 16K of RAM (16384-32767). If all you wanted was 16K, you are finished.

3) If you want more RAM, you must activate the rest of the RAM. Run the following program:

```
POKE 16388,255
POKE 16389,255
NEW
```

This will clear and activate all the addresses to 65536 (64K), except, of course, ROM (0-8192). The activated addresses include 8193-16383 which are not included in any Timex or Sinclair hardware.

4) Now we must test the RAM pack. Use Listing 1 again, but change 49151 to 65535 in line 20.

Some of the same things that plague the TS1500 are also present with the 64K RAM pack. Let's look at some of the good and bad points of the system as a whole.

1) You have available an additional 8K of RAM in the 8K-16K area which is normally not available with Timex/Sinclair hardware. The basic system does not give access to this area; you have to use PEEKs, POKEs, and machine code, but nevertheless it is still there.

2) This 8K-16K area is a nice place to store machine code. You do not have to lower RAMTOP. You could store machine code in a REM statement, and then after your program is loaded from cassette it can be POKEd into the 8193-16383 area via a FOR-NEXT loop.

3) The problem with the display file running into 32676 still exists; however, with so much RAM it is possible to push this file beyond 32676 so larger programs are possible.

4) For data storage the 64K RAM pack is a must.

5) The same checks performed on the TS1000/1500 by Listing 2 are applicable to the 64K RAM.

"And this is Dr. Johnsen, the inventor of the dot matrix printer."

The Intercontroller

Lawrence A. Kelly

The Intercontroller HV bus. $99.95. Intercomputer, Inc., PO Box 90, Prudential Center, Boston, MA 02199.

In *SYNC* (3:4) I reviewed the Votem—a device which converts continuously changing voltages from sensing transducers, such as a temperature probe or photocell, into digital information which the computer can understand.

The limitation of the Votem which I noted then was that, though it could monitor an outside event very accurately, the Votem by itself could do nothing about it. As a possibility for control, I mentioned the Byte-Back Control Module. Another option is offered by Thurnall Electronics which has a relay driver box or transistor driver that can be connected to the computer via the I/O port. Both of these devices would require a modest knowledge of electronics to be successfully implemented.

The Intercontroller gives us a third option that controls outside events, that is reasonably priced, and that is extremely easy to use with completely Basic programs.

To get the Intercontroller working, requires four steps:

1) Connect the cable to the expansion port of computer.

2) Plug in two AC cords (Plug the Intercontroller power strip into a wall outlet. Plug at least one device, perhaps a desk lamp, into the Intercontroller.)

3) Use the direct Basic command.

POKE 8192, 1

If you have plugged your lamp into the first outlet on the Intercontroller, the LED should glow and your light will come on.

4) Write a Basic program that will take advantage of the timing and decision-making capabilities of the computer.

The four grounded AC outlets on the Intercontroller HV Bus accept plug-in devices that will draw up to 9 amperes current. (The Thurnall unit will not stand half as much current, but it will take 230V.) The Intercontroller is only stated as being rated for 115V.

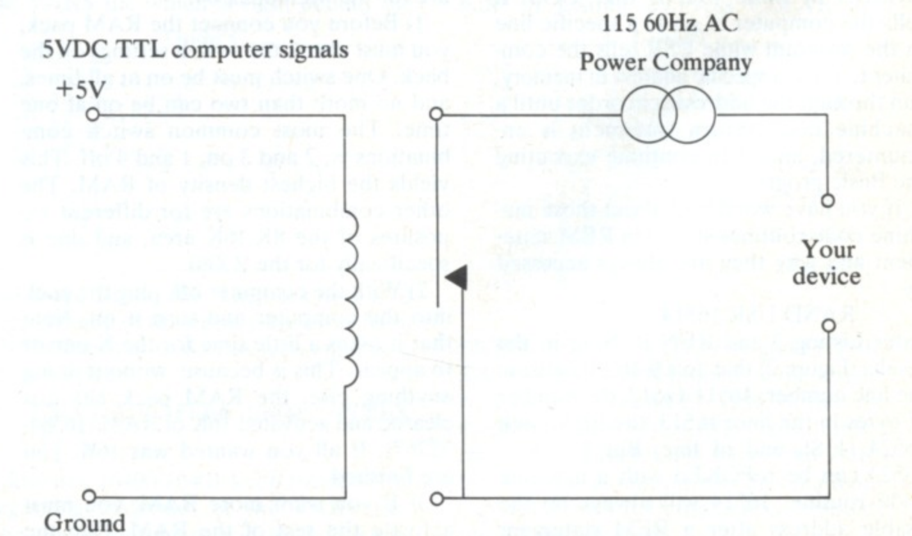
The Intercontroller is not extremely sophisticated as far as a D/A since it is a binary event. It just switches on or off. The main problem in doing this is keeping the 115VAC away from the 5VDC TTL (Transistor Transistor Logic) levels of the computer. A schematic for the Intercontroller was not available, but a typical relay type circuit which shows how this can be done is found in Figure 1. The coil on the computer goes high (1), causing the switch to close in an SSR (Solid State Relay) or EMR (Electromechanical relay), thereby activating the AC high voltage circuit.

The documentation (other than technical) with the device is clear. Several approaches to timing and ideas for implementation are presented. One of the things I plugged into the strip was the TV set. There was no need to keep the screen display going while the computer was just timing so I had the unit turn the TV off while timing and turn it back on when it needed to display something. My daughter thought this was truly elegant—a computer that was smart enough to turn itself on and off!

The device is very nicely made. But one warning should be given. In order to be so "friendly" to use, the device is addressed at memory addresses in the unused 8K area above the Sinclair ROM. This is fine if you understand that you cannot plug in other devices that will want to use this memory space. The low order bit is all that is used to switch the socket on. Thus, POKEing a 1 (00000001b) turns the outlet on; POKEing a 2 (00000010b) shuts it off; POKEing a 3 (00000011b) turns it on, etc., up to POKEing 255 (11111111b) which, of course, turns it on.

The unit could be a useful asset to many automation projects in the home, office, or laboratory. Compared to equivalent control systems, the cost is nominal.

Figure 1. Typical relay to allow TTL to drive line voltages.



Lawrence A. Kelly, PO Box 484, Morris Plains, NJ 07950.

ZX/TS ROM and RAM Addressing

V. B. Rice

I decided to investigate how to wire around the strange effects of PEEKing and POKEing outside the 16-32K address range.

ROM/RAM Immaterial

As a result of my article "Your Timex-Sinclair Can Become a Remote Terminal" (*Sync* 2:6) a number of readers raised questions concerning the peculiarities resulting from attempting to use memory-mapped I/O devices mapped to addresses not affected by the use of a RAM pack. I have seen many references to the strange effects of PEEKing and POKEing at addresses outside of the 16K to 32K range on a Sinclair, so I decided to spend a rainy afternoon investigating the various phenomena, and, more to the point, how to wire around these effects.

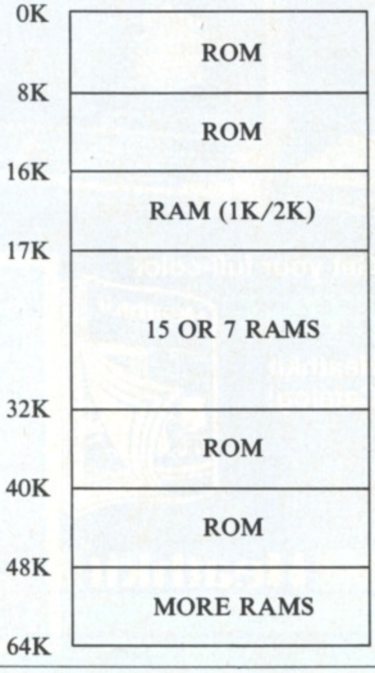
Various Standards

The address lines on the Z80A CPU are numbered from A0 to A15. These lines can be compared to the equivalent bits in a two-byte address: A0=1 when 'on', A1=2, ... A10=1024, A11=2048, A12=4096, A13=8192, A14=16384 and A15=32768. Thus, when all address lines are high, you have a maximum value of 64K (or 65535), and, when all are low, you have an address of 0.

Note that above I referred to all address lines being 'high'. With normal TTL (transistor to transistor logic), an active line is normally low voltage (or grounded), and an inactive line is high (or +5 volts). The reverse is true for the data and address lines on the Z80A CPU for some reason known only to the designers, so a '1' is high or +5 volts, and a '0' is indicated by a grounded or low voltage line.

Since the 1K ZX81 is the only machine on which I have had actual hands-on experience, the remarks on the ZX80 and the TS1000 are assumptions based only on what I have read about those computers, and from examination of their circuit diagrams. Where they differ drastically from the ZX81 I will mention them specifically.

Figure 1.
Map of "echoes" in the ZX/TS address space.



The Strange Effects

As an initial demonstration, detach any add-ons from the CPU so that you have a plain unadorned 1K or 2K machine. Now enter the following little program:

```
1 REM ABC
```

If you now PRINT PEEK 16514, you will get a 38 printed on the screen. This is the value of the A in the REM. If you then PRINT PEEK 49282 (32K off the start of the REM and well off the end of the on-board RAM), you will find the same value printed. If you then POKE 49282,39 and LIST your little program, you will discover that the first statement is now 1 REM BBC. What you have done is to POKE a value way off the end of your memory but which changed your actual RAM contents anyway.

Another interesting demonstration is to PRINT PEEK 0, PEEK 32768 and the two answers will be the same. PRINT PEEK 0, PEEK 8192 has the same effect. To make things even more confusing, you will find echoes of the REM statement above every 1K throughout the addressing space (2K probably for a TS1000), except in those spots where the 'ghost' ROMs are hiding. The resulting map of the various echoes is illustrated in Figure 1.

The Causes

Apparently the designers of the original Sinclair decided that a maximum 16K RAM would be added, so the entire address space above 32K was ignored for the sake of a simplified internal lay-

V. B. Rice, 119 Exeter Rd., Ajax, Ont., Canada L1S 2K4

out. Additionally, since the RAM started at 16K and the ROM was only 4K or 8K long, the space between the end of ROM and the start of RAM was ignored as well.

Electronically speaking, line A15 is connected only to the keyboard and nothing else inside the Sinclair. This results in addresses above 32K being identified by the on-board chips as the address less the value of A15 (i.e., the address less 32K). Thus the duplicated RAM addresses. When the ROM is accessed (when A14 is low producing a resulting address less than 16K), A13 is ignored so that an address over 8K is interpreted as the address minus the value of A13 (or 8192).

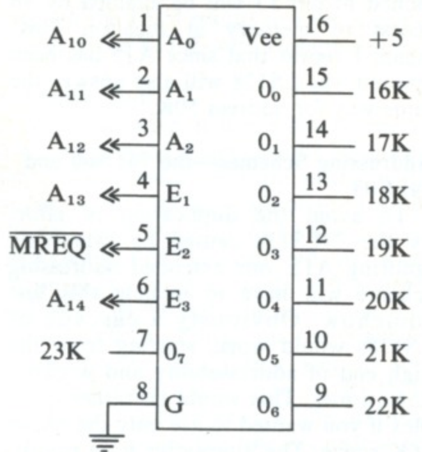
Eliminating the Effects

Since all these ROMs and RAMs are running around the address space, how do you get rid of them so you can do some useful work? Luckily for us, the designers of the Sinclair gave us a couple of very potent levers to work with (except on the ZX80 they forgot one).

RAMCS (pin 2 on the top of the back edge connector) enables or disables the on-board 1K or 2K RAM. If this line is pulled low, either by the Sinclair or by some of your wiring, the on-board RAM is enabled and will answer. The 16K

Figure 2.
A 74LS138 wired up to respond to address in the range of 16K-23K.

Note: This circuit will also respond to addresses above 48K since address line 15 has been ignored.



RAM pack connects this line directly to +5 volts, permanently disabling the on-board RAM. This is why adding a 16K RAM pack gives you 16K of RAM, not 17.

ROMCS (pin 23 on the lower edge connector) will engage or disengage the ROM chip. This pin should be very cautiously approached because tying it high at an inappropriate nanosecond will result in the Sinclair's brain disappearing suddenly.

This is the line they unfortunately forgot to bring to the outside world on the ZX80. You can provide your own (if you have sufficent courage and faith) by soldering a jumper from the edge connector lower contact 23 (if there is one) to the ROM pin 20. This pin (ROM 20) should be connected originally to IC6 (a 74LS157) pin 7. Carefully cut this trace and bridge the gap with a 1.8K resistor. I give no guarantees on this, having never tried it, but I have seen it suggested in several places by people who should know what they are doing.

Addressing Schemes—The 74LS138

Enough of explanations, and on to some specific details for adding new chips. The building block of the addressing schemes which I use is the versatile 74LS138, Figure 2 shows a 74LS138 wired up to select chips at addresses 16K to 23K in steps of 1K. The chip select lines coming out of this chip are controlled as follows:

Pins 4, 5, and 6 are three enable chip lines. For the 74LS138 to be enabled, pins 4 and 5 must be low, and pin 6 high, all at the same moment. Although this may sound strange and complicated superficially, it is actually very useful because it eliminates the need for external decoding chips (in simple circuits). Remembering the 'values' of the various address lines, we can see that if A13 is low and A14 is high, the resulting address must be between 16K and 23K. (In keeping with on-board Sinclair technology A15 is ignored in this diagram).

Pins 1, 2, and 3 are designated as A0, A1, and A2 on the LS138. This designation pertains only to the pins on the LS138 and is in no way to be confused with the address lines on the Z80A of the same names. When the LS138 is enabled by the correct values applied to the enable pins, the values of the lines connected to the address input pins 1, 2, and 3 are added together. (In binary again: A0=1, A1=2, A2=4 when high, so a maximum of 7 is possible, with a minimum of 0, hence the 8 output labelled O0 to O7). The output pin with the designation of the binary total of the address inputs is then pulled LOW until the LS138 is disabled again.

Referring again to Figure 2, if A13 is low (8192=0), A14 is high (16384=1) and MREQ is low (memory access requested). The LS 138 is enabled.

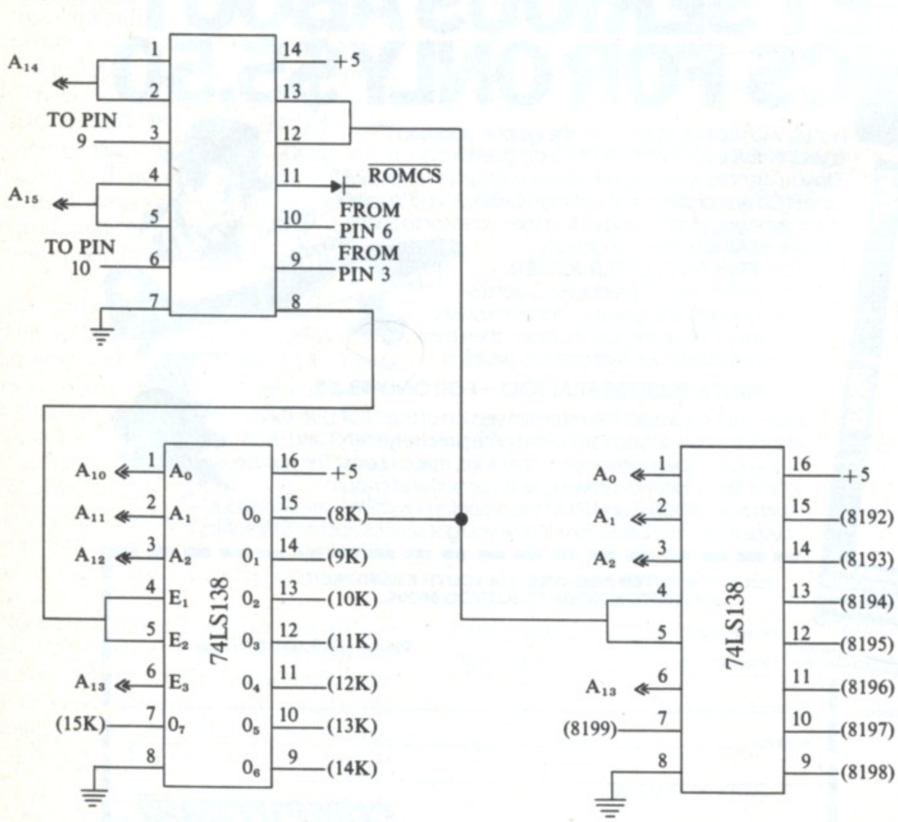
If CPU line A11 is high (2048=1), and lines A10 (1024) and A12 (4096) are low (=0), then the LS138 sees its address inputs A0=0, A1=1 and A2=0, for a binary value of 010, or decimal 2. It will then pull its output 2 (pin 13) LOW in response. Thus, whatever chip is attached to pin 13 will be enabled by an access of memory at location 18K. Again I repeat that since A15 has been ignored, the LS138 will also answer the same way for address 50K.

Addressing Schemes—the 74LS00 and 74LS08

To avoid the duplication of effort by the 74LS138 caused by everybody ignoring A15, our extended addressing scheme will have to include this line somehow. Obviously a cascade of LS138s would work, starting from the high end of addressability and working downwards. This would be rather complex if you wanted to use only the 8K to 16K space. The alternative is to pre-filter the unwanted addresses by using simpler AND and NAND gates (which are also considerably less expensive). The 74LS00 NAND gate consists of four 2-input sections, each of which will pull its output line LOW only when both of its inputs are HIGH. The 74LS08 AND gate is the exact reverse: the output is LOW unless both inputs are high for each section.

Figure 3. Addressing scheme for the range 8K-15K.

Note: The second 74LS138 addresses 8192-8199 in 1-byte increments. A13 is used as E3 on the second 74LS138 merely because it had to have been high to enable the first LS138; +5 would serve equally well.



How would we use these in Figure 2? Remember that we want both A13 and A15 LOW to address between 16 and 23K. The method of eliminating two HIGH address lines is to use 3/4 of a 74LS00 as follows:

1) One of the lines (A13 in this case) goes to BOTH inputs of one circuit, with the output of this circuit going to the input of a third circuit on the same LS00.

2) The other address line (A15) goes to BOTH inputs of the second circuit on the LS00, with the output going to the remaining input of the third circuit.

The output of the third circuit will now be pulled low only if both the original inputs were LOW (try it out on paper).

Connecting the output of the third LS00 circuit to the E1 of the LS138 (pin 4) of Figure 2 (replacing the A13 on there in the diagram) will force the LS138 to answer only to addresses between 16 and 23K.

Figure 3 shows the LS00 and LS138 circuit to address the space from 8K to 15K. Reversing lines A15 and A13 in Figure 3 will address 32K to 39K.

Figure 4 shows how to address 40K to 48K. The rest I will leave to the reader to design if he needs them. The idea is exactly the same.

The 74LS08 chip is used to combine two output lines from the LS138 in order to enable 2K devices such as a 2016 RAM chip, or multiple enable chips like the 8251 USART. Figure 3 shows how one 74LS138 is used to access the 8K to 16K space, and then a second is used to close the range to the byte level from 8192 to 8199. Addresses 8192 and 8193 can then be combined by a 74LS08 to activate an 8251 control or data port.

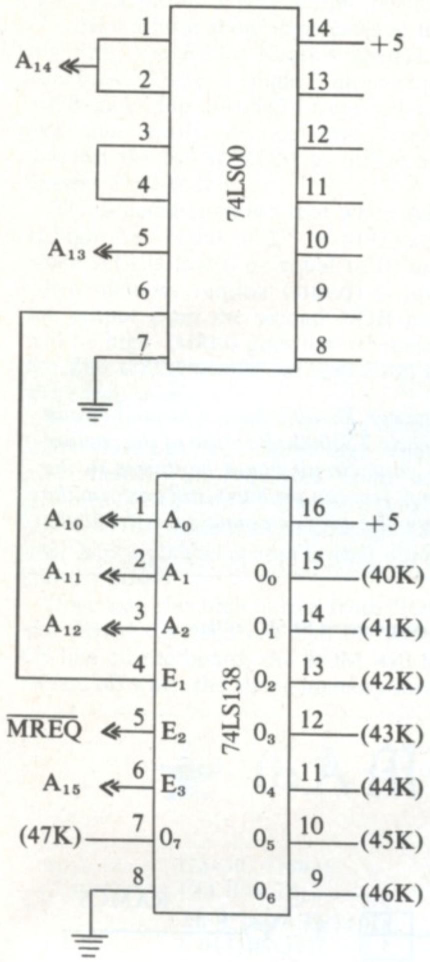
RAMCS and ROMCS

When you add memory chips from 16K upwards you will have to take the on-board RAM into consideration. If the RAM being added is in the form of 2K or 1K chips, it is obviously worth while to use the on-board RAM as well by connecting the first (16384) output pin of the LS138 (pin 15) to the Sinclair RAMCS line. If you have a TS1000 with 2K on board, you will have to connect the LS138 pins 15 and 14 to the inputs of an LS08, and the output from that circuit to RAMCS. Use the subsequent LS138 outputs to enable the new RAM chips. If you are adding 16K chips, the easiest solution is to connect RAMCS to +5 and forget the on-board RAM.

ROMCS is a somewhat different story. If the new chip addresses are from 16K to 32K, ignore this line entirely. The same goes for 48K to 64K. For the other addresses, the simplest method is to invert each chip select line from the LS138 which you intend to use (via a

Figure 4.
Addressing circuit for the range 40K-47K.

Note: The 74LS00 ensures that A14 is low and A13 is high. Compare with Figure 3 to see the differences in the 74LS00, pins 3-5.



74LS04), then connect the LS04 output to a small diode (anode end) and connect the cathode to ROMCS. (The cathode end of a diode normally has a color band around it, usually black). If you omit the connection to ROMCS, the ROM will answer as well as your new chip, resulting in unexpected data being passed back to your program. If you omit the diode, the high value from the LS138 connected to the ROMCS when the LS138 is not being enabled going through the LS04 results in a permanent enable of the ROM, with strange results.

A Simple Tool

As a final note, it is possible to check out your address select circuits by building a very primitive logic probe with which to observe the state of the LS138 (or any other one-shot TTL) lines. Take the smallest LED you can find, and wire the long pin onto a handy +5 source and the short pin onto the select line you want to check. Power up the Sinclair and turn down the lights so you can see the LED flash, then type in PRINT PEEK xxxx where xxxx is the address that should select the line connected to the LED. If it flashes, you are OK. A short loop program that runs through the PEEK continuously will light the LED somewhat better. If the LED does not flash, hook it across +5 and ground to make sure it works, and check the wiring on the LS138 again. If it stays lit all the time, something is wrong with the enable wiring on the LS138 so it is enabled continuously.

Checking address and data lines this way is pointless since the keyboard scan routines in the ROM keep activating these lines constantly.

It is also possible to make up a circuit around a latching type chip (like a 4013) with a manual switch and connect that between the WAIT line and ground to 'single-cycle' the Sinclair. This will have the effect of 'freezing' each signal, but the thousands of machine cycles involved in a single PEEK would wear out your switch-pushing finger pretty quickly.

I hope that this discussion has helped to explain why the Sinclair reacts the way it does so that some of you will be able to add more good things to an already fine computer.

"I asked you to write, 'I will not be tardy,' 500 times. A computer print-out is unacceptable."

A WORD OF CAUTION: *Any hardware project for your computer must be approached with extreme caution.* **SYNC** *cannot be responsible for any problem that may arise from attempting hardware projects. Obviously, any damage to your computer can be costly in time and money.*

The Great RAM Rescue

Randall S. Glidden

The Unused 1K-2K Space

So, do those of you who have upgraded your ZX81/TS1000 to 16K ever feel sorry for that unused 1K-2K RAM, sitting unloved and unneeded inside your computer? Why not put it to work for you by wiring together this little circuit? It will convert your on-board RAM into a handy little place to put machine code programs, cost under $10, and, if you are handy, take about 15-30 minutes to wire up. Here, nestled away, will be a memory location unaffected by LOAD, NEW, and (with a little switch) MC induced crashes.

The on-board RAM begins at address 16384 (4000h), and it is deactivated when you plug in an external RAM pack. This is done by putting the +5 volts to the RAM chip select line (RAM CS), which turns off (deselects) the internal RAM chip. (The RAM chip is active or *selected* when RAM CS is *low,* i.e., close to 0 volts.) You can use this internal RAM at any address you desire, simply by making RAM CS low whenever a certain range of addresses is called for.

For machine code programmers a convenient place to use in the Sinclair memory map is the area 8192-16383, since it is not affected by LOADing or the NEW command. Ordinarily this area is occupied by the ROM operating system which selfishly repeats itself there out of its "normal" position between 0 and 8191. This was done apparently for simplicity of decoding—the Sinclair logic chip makes ROM CS low (i.e., turns on the ROM) whenever the A14 address line is low (therefore from addresses 0-16384 and again from 32768-49151).

Opening the 1K-2K- Space

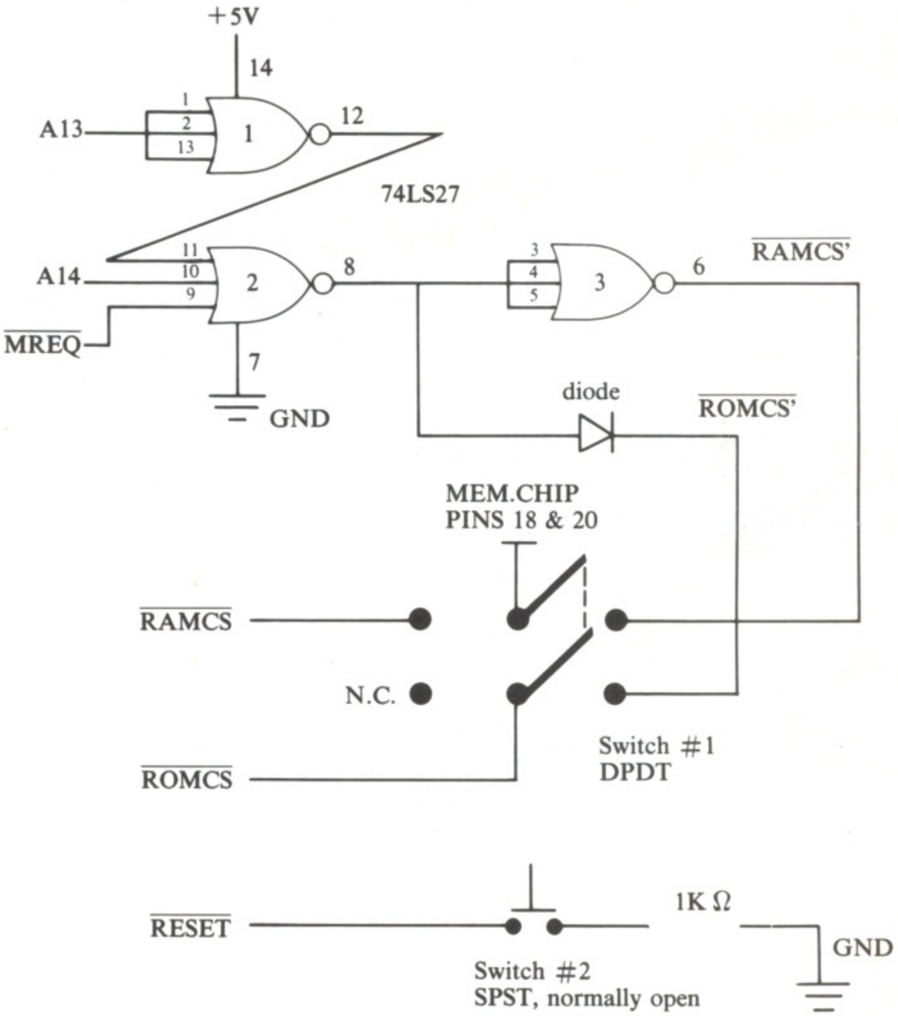
To free up some of this wasted space all you have to do is put ROM CS high whenever some of these addresses are

Randall S. Glidden, M.D., 185 Chiswick Rd., Brighton, Mass. 02135.

Author's Addendum

It has recently come to my attention that newer TS1000s have a printed circuit layout that differs from that of the ZX81 and early TS1000s. Because of this, before you attempt this modification, be sure that your circuit board conforms to the pattern shown in the figure. If it is not the same, you can probably still perform the "RAM rescue," but you will have to identify the proper connections from the appropriate ROM or CPU pins as noted.

Figure 1. Schematic for RAM/ROM decoder and RESET switch.



needed, thus "turning off" the ROM in the same way a RAM pack turns off the internal RAM. With the ROM out of the way, we are then free to use this memory area.

The circuit shown in Figure 1 utilizes this principle in a very simple fashion. (To learn a little about logic gates to follow this, see Radio Shack's *Engineers Notebook*.) Using a single 3 input triple NOR gate chip (74LS27 or equivalent), you can easily decode and select/deselect the RAM/ROM to utilize addresses 8192-16383.

The schematic shows that when A13 is high (i.e., equal to 2^{13} or 8192) *and* when A14 is low (i.e., equal to 0) *and* when memory request (MREQ) is low, the output from the second NOR gate will be high. (MREQ goes low whenever the Z80 CPU "wants" to read from or write into memory.)

Looking at it another way, the output from NOR gate #2 will be high whenever memory is requested from addresses 8192-16363 (or addresses 40960-49151, since the relationship A14 low, A13 high also occurs in that upper address area).

If we take this high output from NOR #2 (ROM CS') and put it to the ROM CS line on the board, the ROM will be turned off when the above memory locations are addressed. The diode shown in the schematic is necessary to prevent a low from enabling the ROM at inappropriate times, but it allows a high to get through when we want it.

We can take this same high output, put it through the third NOR gate (thus inverting it to a low—RAM CS') and use it to enable our on-board RAM via the RAM CS pin (pin #18). You must first remove the RAM chip and bend out the RAM CS pin so that it no longer makes contact with the disabling +5 volts coming from your external RAM pack.

What you have after all this is your on-board RAM chip selected (i.e., ready to send or receive data) and your ROM deactivated whenever memory locations 8192-16383 or 40960-49151 are addressed. You have decoded a lot more memory area than you need for your 1K-2K RAM chip, but they are usually unused areas anyway.

Be aware, however, that some peripheral devices use the 8K-16K area in their hardware design, so in those cases this decoding scheme could give you some real problems. Consult your user's manual if you have any extra hardware (other than a 16K RAM pack) before you jump into this modification. If it uses *any* memory locations in that area you had best leave this project alone.

Another theoretical note: you will note on the schemactic that the signal to the RAM CS pin (pin 18 on a 1 or 2K chip) from the third NOR gate also goes to pin 20. This is the READ enable pin which can be held low, whether the RAM chip is being READ from or WRITTEN into. (There is also a WRITE enable pin on the memory chip which permits data to be written into RAM when low and read from RAM when it is high, as long as the READ enable pin is held low at all times.) Since the Timex/Sinclair circuit board ties together pins 18 and 20 (and thus to the RAM pack's +5V), you have to disengage both of these pins from the socket and wire them together to your RAM CS' signal.

The optional DPDT switch (switch #1) will allow you to switch back to the on-board RAM at its usual memory location when you are not using the 16K RAM pack. If you *never* want to use the on-board RAM without the RAM pack, then just wire the circuit in without the switch.

Adding a Reset Switch

If you have done much MC programming, you are probably painfully aware of how easily MC programs can crash your computer. Even if you have your MC program in the 8192-16383 area (where it is safe from NEW) and you crash the computer and unplug the power, good-by MC.

A simple way around this uses the RESET pin present on the Z80 CPU. This pin (pin #26) is held high at most times. If you temporarily pull this pin low, the CPU will stop what it was doing and will go back to the beginning of the ROM (address 0), just as if you had just turned the power on.

During a crash, making RESET low will effectively get the computer going again, but, since the power was not interrupted, the MC program that you had in the 8192-16383 area will remain perfectly intact (and ready to crash your computer again!).

By connecting a momentary SPST switch (switch #2) and a 1K ohm resistor between RESET and ground, you will have a handy reset button to use in the event of a crash. Remember, however, that if you press reset at any other time you will wipe out your program above 16384, just as if you had pressed NEW.

Construction

Read these instructions all the way through before you actually begin so that you know what you are going to do.

1) Parts and tools.

You will need the following items:

1. Small Phillips head screwdriver
2. A 15-25 watt fine tip soldering iron.
3. Solder.
4. 30 gauge wire-wrap wire and wire wrapping tool.
5. 1 triple, 3 input NOR gate (74LS27). Available at most electronics supply houses, but *not* at Radio Shack.
6. 14-pin DIP socket (solder tail type).
7. 1 switching diode (1N914, R.S. #276-1620 or other fast switching diode).
8. DPDT switch (R.S. #275-403 or equiv.).
9. SPST submini momentary, normally open switch (R.S. #275-1571 or equiv.).
10. 1 1000 Ω ¼ watt resistor.

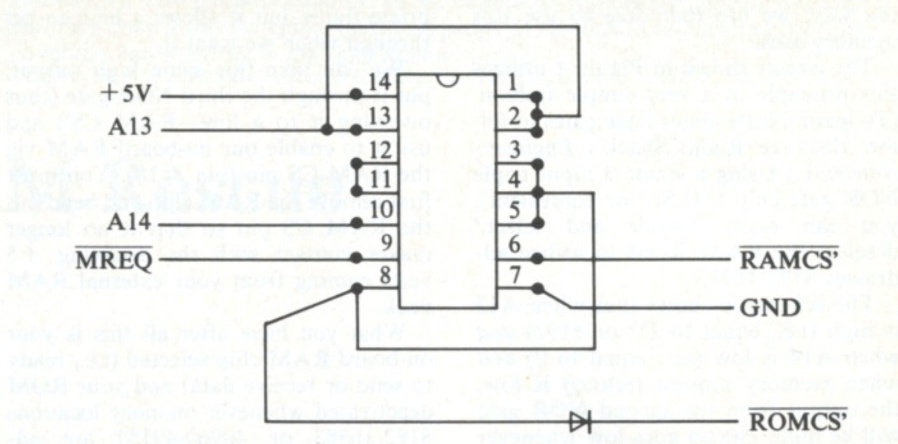
2) Wire the circuit.

Following the circuit diagram (Figure 2), make the appropriate connections to the 74LS27. This is best accomplished by making them directly to the DIP socket. For connections that will be made to the Sinclair circuit board, solder 2-3 inch wire-wrap wire to the appropriate pins of the socket. Strip about ⅛" insulation from the ends of wires that will be soldered and about ¾" from the wire that will go to the memory chip.

The diode can be soldered either to pin 8 of the socket or directly to the

Figure 2. 74LS27 socket wiring diagram.

Note: Pin positions as viewed from underside of socket.



ROM CS hole on the board—your choice. Make sure you have it oriented in the right direction—usually the banded end is the anode, i.e., the end that connects to the board.

Put the chip in the DIP socket, making certain that it is in right side up (i.e., Pin #1 in socket #1).

3) Remove the circuit board.

Remove the five phillips head screws from the base of the computer (three are under the lower and upper left rubber pads). Note that there are both long and short screws and make note of which ones go where. Remove the back and then the two screws holding the board in place. Carefully turn the board over and gently remove the keyboard tails. (Do not rip them!)

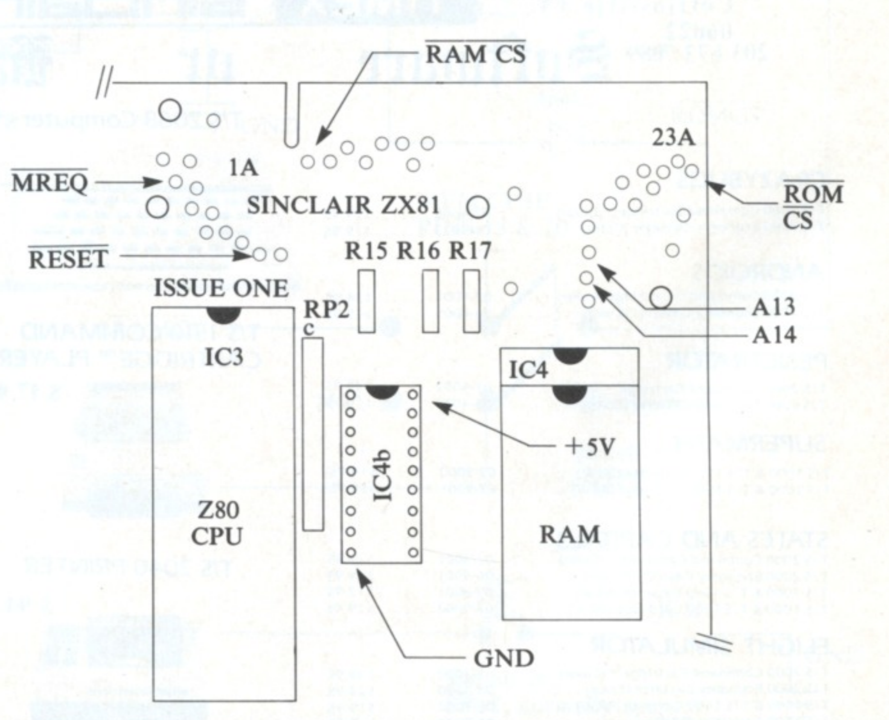
4) Locate landmarks.

Match up the locations of the appropriate holes (PCB feed throughs) on the board as shown in Figure 3. Also look at the picture in your Sinclair Basic Programming book (same pictures in ZX81 and TS1000) and figure out which little black box is the RAM chip and which is the CPU.

If your computer is a ZX81, you may have two small RAM chips instead of one larger chip. You will still be able to do this project, but you will be a bit more cramped for space when it comes to placing the 74LS27.

Once you have located all the holes, make sure they are correct by checking continuity (preferably with a low-power ohm meter) between the holes and the

Figure 3. PCB connections diagram.



appropriate pins of the CPU or ROM as follows:

Line	*Chip*	*Pin*
RESET	CPU	26
A14	CPU	4
A13	CPU	3
MREQ	CPU	19
RAM CS	RAM	18*
ROM CS	ROM	20
+5 volts	ROM	24
Ground	ROM	12

* or RAM pin 8 if two smaller RAM chips present.

5) Solder the appropriate wires to the board.

Use small amounts of solder to prevent solder shorts. Check both sides of board after each connection. Keep the leads of the diode short and insulate them with tape. Solder +5V last.

It is a good idea to power up your computer after each connection and see if a K cursor still appears (you do not need the keyboard plugged in). If any shorts are present, you will get some squiggly lines or other such garbage on the screen. Desolder your most recent connection and try again if this happens.

6) Connect 74LS27 to RAM chip.

Remove the RAM chip from its socket by carefully prying up each end. (Do not bend the pins!) Once removed, bend out pins 18 and 20. (If you have 2 little RAM chips, remove them and bend out pin #8 only on each.)

Now wire-wrap the RAM CS' wire from the 74LS27 to pins 18 and 20 (or pins #8—see above). Replace the RAM chip(s) in the socket(s) with the bent out pins sticking out sideways.

If you are using the DPDT switch shown in the schematic, make the connections as shown.

Before you actually wire it in, find a spot to mount your switch that will not get in the way of things when you are done. Since I have my computer in a large enclosure, I did not actually mount the switch on a "stock" chasis. The upper right corner of the front cover is probably a good location.

7) Reset button.

If you want to mount your SPST switch, find some out of the way place on the case, drill a hole, and mount it. Solder the 1K ohm resistor to one pole of the switch after trimming its leads as short as possible. (Insulating these leads with some electrician's tape is probably a good idea to prevent shorts.) Connect the switch to the ground and RESET holes on the board, with the 1K ohm resistor wired in series as shown in the schematic.

8) Affix the 74LS27 assembly to the board. One easy way to do this is to tape it to the board, upside down, using double-sided tape or a small dab of rubber cement. This will keep the chip from bouncing around inside your computer, which would cause all kinds of mischief.

Testing

I would advise testing before you put your computer back together:

1) Reset button.

This is easy. A brief push of the button should give you a blank screen followed by the return of the "K" cursor. If a program was in memory, it will be gone.

2) New RAM area.

If the DPDT switch was used, switch it so that the RAM chip is connected to RAM CS'. Plug in your 16K RAM and enter this program to test it:

```
10 FOR N=8192 TO 10239*
20 POKE N,0
30 PRINT PEEK N;
40 NEXT N
```

* use 9215 if 1K RAM

This will POKE zeros into each memory location and verify it PRINT-PEEKing each byte. You should get about 2½ screens full of zeros with a 2K RAM. (Since the entire 8K block from 8192 to 16383 is decoded, you will also get zeros if the space between 10239 and 16383 is PEEKed.)

If you are using a 32K RAM pack, you probably should POKE RAMTOP no higher than 40960 (the first 24K). Since our one-chip decoder repeats itself in half of the second 16K block (addresses 40960-49151), you may get into a data bus conflict if you try to use that area. *Some* RAM packs *may* work in this area however—try and see.

Now put the DPDT switch into the other position, remove your RAM pack, and turn on the computer. It should now work just like it used to without the RAM pack.

Trouble shooting

Recheck your connections! There is really little else that can go wrong other than wrong/bad connections and solder shorts. Did you make *all* of the right connections to the board? Is the chip or diode in backwards? If the computer will not work *at all*, then unsolder your connections one by one, testing the computer after each step. Write me if even the above does not help.

If you succeeded with this little project, you now have a sequestered place to put the machine code, and also a way to keep it there after it crashes your computer. You will also have learned a little bit (byte?) about the ZX/TS circuitry and have experienced some of the more intangible rewards of becoming a "computer hardware hack."

Forth

Robert B. Trelease, Jr.

Do any of the following software "dilemmas" seem familiar?

Jerry wants to write games programs which perform faster than his Basic versions, but he cannot quite get the hang of calling and using extensive machine code subroutines. He has heard, however, that he can get a tape-loading compiler-based high-level programming language which runs extremely rapidly . . .

Bob has just obtained one of those new analog-to-digital converters capable of sampling at 200 KHz with Sinclair/Timex computers. He knows that machine code subroutines are needed to obtain the maximum data acquisition rate; however, he wants to speed his evaluation of different A/D converter applications by programming in the most rapidly executing high-level language available . . .

Sara is a professional programmer who is intrigued by the simplicity of her Timex Sinclair 1000 and is using it to develop a college course on computer "literacy" and computer aided instruction. She thinks Basic is "just OKAY," but she wants to add a new language so that she can program in structured, threaded code, interpret *or* compile program statements, and still use machine code subroutines . . .

If such situations seem somewhat familiar, you have probably personally experienced some of the shortcomings of working with the *interpreter* Basic operating system of the Sinclair/Timex computers. As elegant as the Sinclair Basic ROM is, execution of commands can be relatively slow, especially in SLOW mode. Machine code routines called from Basic can approach the maximum computational speed obtainable from Sinclair/Timex hardware, as numerous articles and tutorials in *SYNC* have demonstrated.

However, there is now an alternative programming language and operating system available which runs many times faster than Basic, allows machine code subroutines, streamlines program writing, is capable of being *extended*, interprets *and* compiles code, and is capable of dealing with the aforementioned "sample problems." That language is FORTH.

This article provides an overview of this unique and powerful programming language, a review of the Tree Systems PLURI-FORTH EPROM-resident operating system for the Sinclair ZX81 and Timex Sinclair 1000, and selected references for those who want to do further study.

Robert B. Trelease, Jr., 2313 5th St., Santa Monica, CA 90405.

Photo 1. Instrumentation set-up. ZX81 with Suntronics KB conversion, TS2040 printer, 16K RAM, Computer Continuum 8 channel A/D D/A Board, CC expansion board (shielded) and "special purpose" RAM/function board. Not shown dual Basic/Forth operating systems.

What is FORTH?

Like Basic, Fortran, Pascal, and others, Forth is a "high-level" programming language whose statements or commands are converted into machine ("object") code instructions by the operation of *interpreters* or *compilers* resident in the computer memory. In systems like the Sinclair/Timex, whose Basic is stored in ROM with an operating system/monitor, only an interpreter is used during program execution. Thus, each statement line is individually and sequentially "interpreted" as machine code as it is encountered during RUNning or direct command entry.

Similarly, in Forth systems, program statements can be interpreted and executed directly upon entry. Program statements can also be *compiled* in Forth, allowing the user to run routines pre-converted ("compiled") into machine code.

Threaded Coding

Despite this extra dimension, Forth is considered an *interpretive* language. This combination of interpreted/compiled

Perhaps the most important attribute of Forth is its extensibility. It allows new "commands" and functions to be added with simple programming steps.

operation interacts with another important Forth "characteristic": *threaded coding.*

To comprehend *threaded coding,* we must consider how a high-level language can be constructed. Initially, the language consists of a number of named fundamental subroutines (operations) written in the machine ("operation") code of the system CPU (e.g., the Z80). These subroutines may include mathematical and logical manipulations of numbers in the CPU registers, reading of keyboard character input, and organization of character output to control a display device.

To expand the language or construct a program, we must string together calls to these named subroutines *along with* interposed data bytes. A program in such a language thus represents a *thread* running between a structured list of subroutines. "Threadedness" is not an inherent characteristic of any given programming language—it is a language use technique. We can thus find examples of threaded structure in other languages, such as in Basic programs using numerous subroutine calls. Lawrence Auer has demonstrated several aspects of this technique in "INTERP—The Kernel of Interactive Nuts" (*SYNC* 3:1, pp. 43-47) in which he illustrates how a Forth-like language interpreter can operate.

Threaded coding is a normal feature of Forth programming and primarily accounts for its impressive speed of execution. Threaded programming is enhanced by *"structured"* programming, and Forth is a "structured" language. Although this may not seem logical to casual users of Basic or Fortran, these latter languages are "unstructured" in that program execution is controlled by sequential conversion of *numbered* sequences of statements. Forth programs represent *structures* whose execution depends on the *patterns* of subroutine calls and construction of user-defined "commands" based on previously defined "commands."

Extensibility

Perhaps the most important attribute of Forth is its *extensibility*. The language was designed to allow new "commands" and functions to be added with simple programming steps. In Forth, the most fundamental *WORDS* (the Forth designation for "commands") are constructed as named calls to machine code subroutines, as described for the hypothetical language example above. Expanding the language thus consists of defining new *words* by stringing together data numbers with previously defined *words.* The actual process for creating new *words* is discussed below.

Forth for a given computer is usually provided by the source vendor with a minimum number of fundamental *words* and this "*kernel*" (Forth terminology) serves as the basis for further expansion. The "list" of *words* in the *kernel* along with any created "extension" *words* is referred to as the *dictionary*.

Due to the development of standards for Forth systems over the past decade, most implementations of the language have very similar (if not identical) *kernels* regardless of which CPU is used in the host computer system. This commonality of *kernels* and *word* format means that Forth programs are relatively easy to transport from one machine to another, and new *words* (i.e., extensions) can also migrate between systems. Additional aspects of Forth "standards" and transporting programs between different machines will be considered below.

In programming terms, then Forth is best described as a "structured, threaded, interpretive, and extensible" computer language. Because it is usually supplied in a form that conducts general input/output operations, as well as handling storage and peripherals, Forth often serves as a computer's "operating system." Thus, discussions of the language typically include "operating system" functions.

What's Forth Good For?

At this point, after going over the *theoretical* (frequently pronounced "boring") aspects of Forth, you are probably anxious to find out "What it's good for . . . " A convenient escape would be to say "It's good for nearly every computing application" and go on to the programming examples. However, Forth has had an interesting developmental history, and reviewing some of the highlights will help illustrate some of the power of the language.

Forth was developed over a number of years by Charles Moore, a "programmer's programmer" conversant in a number of languages. During the period in which Forth evolved, Moore was writing programs for industrial and scientific data acquisition and control, most prominently for radio telescopes at the National Radio Astronomy Observatory. Figuring that a very good programmer could account for only one "good" program a year for 40 years of production, Moore desired a tool to speed program writing and execution, increasing productivity. Forth was that tool.

The first "complete" Forth system was implemented at Kitts Peak Observatory, and the popularity and the use of the language for data acquisition and control grew among astronomers. Forth was officially designated as one of the approved programming languages for radio astronomy, and soon other types of users were exposed to its speed and power.

In more recent years, Forth has found its way into a variety of applications. It has been used in large hospital computer systems to control patient data entry and exchange between laboratories, nursing stations, and administration. It has been used to control data acquisition in computerized, portable heart (EKG) monitors. Forth has also been used for game program development at Atari. In the brain research laboratory in which I work, an extended version of Forth has been used for high speed physiological data acquisition and subsequent analysis on a Digital Equipment Corporation LSI-11 based computer system.

As Forth grew in popularity, user groups formed. Perhaps the most important of these is the American FORTH Interest Group (FIG). FIG joined with other groups, including European users, FORTH Inc. (the company formed by Charles Moore and colleagues for producing commercial Forth packages), and software producers, in attempting to establish standards for Forth syntax and dictionaries.

These efforts, as exemplified by the "Forth-79 standard" have lead to the widespread use of a common set of *words* for the kernel. FIG distributes a public domain version of the language which closely resembles that described by Forth-79 standards. The FORTH Interest Group in particular is a very good source for information about standards and language usage. Such information can be extremely useful, given the tendency of software vendors to produce "supersets," containing relatively standard kernels with special extensions (extra *words).*

The Monitor And Forth Stack Operations

When a Forth system is first "started up" (loaded from cassette or disc or "turned on" in a PROM), the system prints OK on the video display to indicate that it is ready. Thereafter, the Forth "monitor" replies OK on the video display every time an operation is successfully completed and the system awaits new input. If certain types of error (such as underflow of the numerical stack) are encountered during input, interpretation, or execution, the system will print an error mesaage on the screen followed by OK. The system usually reports errors such as mathematical stack underflow or overflow, use of an undefined *word*, use of a number in the wrong base, or memory full, but otherwise does not protect the user from making errors in writing a program. This situation will be somewhat foreign to the user of a very "friendly" syntax-checking language like Sinclair Basic.

In Forth, mathematical, logical, and operating system functions are carried out using numbers stored in a general purpose "last in, first out" (LIFO) numerical stack set up in RAM. Under "ordinary" program conditions, the "width" of stack numbers is 16 bits, and memory addresses or 16 bit numbers can be stored. Because the Forth numerical stack is a LIFO buffer array, desired data and control numbers must be loaded in proper order *prior* to the use of a Forth *word*.

In computational terms, this means that Forth uses "*postfix*" or "*reverse Polish notation*" in setting up statements for execution. This type of notation will be familiar to users of Hewlett-Packard calculators who are accustomed to ENTERing numbers before punching in a desired mathematical operation. For those unfamiliar with postfix notation, the following example is offered:

```
Basic                     Forth
PRINT 4+2 <ENTER>         4 2 + . <ENTER>
6                         6 OK
```

In either case, the lines are entered directly from the keyboard, and they are directly executed by the interpreter following <ENTER> (<ENTER> indicating hitting the ENTER or RETURN button on the keyboard). The Basic line shown will print the sum of 4 plus 2 on the video display. In the Forth line, 2 and 2 are placed on the stack, and then these stack entries are added together, with the sum left as the top stack entry. The Forth *word* . removes and prints the top stack entry on the display screen. Both of these lines produce the same result (6 output to the screen) if directly executed on the appropriate Basic or Forth system.

In the Forth system LIFO numerical stack, 4 is entered first. Entering 2 "pushes down" the 4, and 2 becomes the number at the *top* of the stack. In *Starting FORTH*, Brodie compares the stack to a dinner plate dispenser in a cafeteria, with numbers added and removed from the top of the stack as plates are added and removed from the dispenser.

Numbers can be added to the stack one at a time by typing the number and <ENTER>. Alternatively, numbers can be entered one after another from the keyboard, with a space separating each number and <ENTER> ending the line. Forth *words* operate on numbers in the order of their appearance on the stack, with numbers resulting from operations placed on the top of the stack as shown in the following example:

```
Forth                     STACK
1 2 3 4 <ENTER>           4 (top)
                          3
                          2
                          1 (bottom)
```

The column entitled *STACK* graphically depicts the contents of the numerical stack, as if they were "plates" in a "dispenser."

Unfortunately or fortunately (depending on whose side you are on), Forth programmers have chosen another convention for simplicity in representing stack contents. Stack numbers are listed from left to right, with the stack "top" at the right. The above columns would be depicted as follows:

STACK DISPLAY
CONVENTION
1 2 3 4 (top)

This convention is useful for demonstrating the results of mathematical operations.

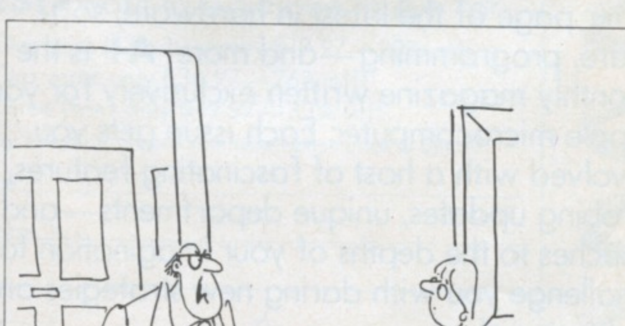
Forth contains a number of *words* which manipulate the numbers on the stack. DROP drops or eliminates the top number on the stack. DUP duplicates the top number on the stack, so it becomes both the first and second value. OVER copies the second number on the stack, entering it again as the new top value. SWAP exchanges the top entry with the second entry. ROT places the third stack value at the top, pushing down the first and second values. Using these and other single and double precision stack operators, complex mathematical manipulations and data transfers can be more easily managed.

Forth Mathematical Operations

As previously mentioned, the Forth numerical stack usually deals with 16 bit numbers. Unless otherwise specified, these numbers are entered and displayed in *decimal*, and they are "*signed*" *integers* ranging from −32767 to 32767.

Normally, Forth systems are thus supplied with *integer* arithmetic, a feature which should be familiar to owners of ZX80 4K integer Basic systems. While 16 bit signed integer arithmetic can be somewhat limiting to the user accustomed to floating point operations, several Forth features help overcome this apparent shortcoming.

First of all, because Forth is extensible, new *floating point words* can be added to allow such operations. Some implementations of Forth currently include floating point *words* as a part of the *kernel*, freeing the user from the task of writing his own new *words*.



"That damn computer thinks it knows it all!"

A NEW BREAKTHROUGH FOR YOUR APPLE™...

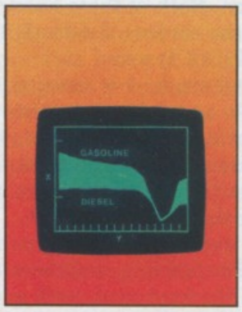
Introducing A+, the Most Significant Advance in Apple History!

It's here! Big, bold and bursting with page after page of the latest in hardware, software, programming—and more! **A+** is the monthly magazine written exclusively for your Apple microcomputer. Each issue gets you involved with a host of fascinating features, probing updates, unique departments—and reaches to the depths of your imagination to challenge you with daring new strategies and adventures!

Here's just a sampling of what A+ is all about:

HARDWARE: From the latest advances in telecommunications and networking...to new ways to augment memory and access a wider variety of peripherals with greater ease...each issue of **A+** gets you involved.

SOFTWARE: What's new on the market and how you can make your system more powerful...plus full reviews and programming information.



UTILITIES AND DIAGNOSTICS: In addition to language utilities, **A+** helps you troubleshoot problems... makes you faster and more efficient...with utilities ranging from record processing to data transfer.

BUSINESS AND FINANCE AIDS: Software for financial forecasting and analysis...general accounting programs...financial news retrieval services...and business graphics so you can see just where your money is going.

GAMES: Take the **A+** challenge—with hoards of innovative brain teasers...from games to help you learn spelling and arithmetic...to games that provide fun and adventure!

PROGRAMMING TOOLS: Speed up program development with Apple text editors, testing and debugging aids and data base management systems. Since your Apple can speak many languages—from Cobal to Basic and Pascal—**A+** gives you a wide range of utilities for each language so you can create your own exciting programs.

HOME/PERSONAL PROGRAMS: covers it all...from basic processing...to helping you draw with Apple graphics...to speed reading...music theory and composition...even a course on cooking with your Apple...and then how to monitor your diet easily and accurately...along with many educational programs to learn math, science, and geography.

DATA FILES: Let **A+** probe new data base management systems so you'll be able to organize, store and retrieve information in ways you never thought possible.

YOUR TURN: Speak up with your ideas and innovations and network with other Apple users near and far.

Plug into Savings of Up to 33% with Our Introductory Offer!

YES! I want to subscribe to **A+** for:

- ☐ One year (12 issues) only $19.97—20% off!
- ☐ Two years only $36.97—26% off!
- ☐ Three years only $49.97—33% off!

Savings based on full one-year subscription price of $24.97.

Name____________________

8H187

Address____________________

City____________________

State__________ Zip__________

Check One: ☐ Payment enclosed. ☐ Please bill me.

Please charge my:
☐ MasterCard ☐ VISA ☐ American Express

Card No._______________ Exp. Date______

Would you like to receive special offers from qualified users of our mailing list? ☐ YES ☐ NO

MAIL TO: **A+**, P.O. Box 2964, Boulder, CO 80322

Forth systems use ASCII code in representing and storing characters.

Secondly, Forth normally supports 32 bit (double precision) and 64 bit unsigned arithmetic by employing *words* defining 32 or 64 bit stack entries and operations. If a normal signed stack number is depicted as a 16 bit "byte," a 32 bit entry can be depicted with the least significant "byte" at the top position and the most significant "byte" in the next position down. Separate Forth *words* are used to specify whether mathematical operations on the stack are *expecting* 16 bit signed, or 16, 32, or 64 BIT numbers.

This utility for switching between number types has disadvantages as well as advantages. The principal disadvantage is that it requires the user to keep track of what kind of numbers he is using at all times. For example, if you enter a 32 bit number and perform a 16 bit multiplication (* in Forth), the product of the most significant "byte" and the least significant "byte" will be left on the top of the stack as a 16 bit number. In a general sense, this demonstrates an important "feature" of Forth—for many operations, it does not protect the user from certain types of data processing errors.

For 16 bit signed numbers, Forth typically supports only multiplication, division, addition, and subtraction. To the Sinclair/Timex Basic user who is used to floating-point trigonometric functions and exponents, this may seem like an incredible shortcoming. However, as mentioned above, the expansible nature of Forth allows the user to add functions as desired by combining previously defined *words* and numbers.

In representing 16 bit single precision numbers (variables) in describing words or stack operations, the symbols n1, n2, n3, etc. are used. For representing double precision numbers, the symbols d1, d2, d3, etc. are used. Unsigned numbers are represented by u1, and so forth.

Forth systems also contain some provision for changing *number base* for input and display. This feature can involve a *word* like HEX, which shifts the system to hexadecimal notation. A *word* like HEX can be very useful for constructing *words* in machine code, as discussed below.

Forth systems use ASCII code in representing characters. Characters and character strings are usually stored by representing them as ASCII code numbers (in 8 bits) on the stack. A string is stored as a list of ASCII numbers with an additional number specifying how many characters are in the string. Special Forth *words*, such as ." and " tell the system that the character input is for output to the screen or storage so the interpreter does not try to use it as a *word* or *words* specifying an operation.

Forth Loops

Like Basic and other high level languages, Forth supports program loops. The following example illustrates this capacity:

Forth	*Basic*
10 FOR I=1 TO 10	11 1 DO I . LOOP
20 PRINT I;	
30 NEXT I	

Either "operation" prints the numbers from 1 to 10 on the video display screen. In the Forth line, two numbers (11 and 1) are entered on the stack to set the count limits for the loop. The second (top) stack entry sets the starting point for the count. "I" is the Forth *word* which represents the standard count increment variable, comparable to the count variable "I" set up in the Basic statement. Look counts include the last integer before the upper count limit.

The Basic and Forth statements are not exactly comparable, inasmuch as the Forth line can be executed directly from the keyboard following <ENTER>, and the Basic statements must be RUN as a program. In some Forth systems, DO LOOPs can be used only in *word* definitions, which are described below. The format for loops in Forth can be shown as:

n1 n2 DO *words* & numbers LOOP

where n1 is the upper numerical limit, n2 is the starting number (index), and the *words* and the numbers between DO and LOOP specify the operation to be performed. Loops which count down can be performed using the *word +LOOP in the following* fashion:

n1 n2 DO *words* & numbers
increment +LOOP

where increment is a positive or negative integer which defines the size and direction of the count increment or decrement.

DO loops can be nested, like FOR-NEXT Basic loops. Forth DO LOOPs are "definite," in that execution is given definite numerical limits. Another type of "indefinite" loop is supported by Forth is:

BEGIN wordname flagword UNTIL

This executes the *word* designated by wordname as long as the flag tests *false* or 0. When the flag designated by the flagword becomes true or non-zero, loop execution stops.

Defining New Words

As previously indicated, much of Forth's power comes from its extensibility, or ability to add new *words*. New *words* are created using the Forth *words* : and ; in the following fashion:

: wordname *words* & numbers;

For example, the following line creates the new *word* COUNT which prints the numbers from 0 to 9 on the video display screen:

: COUNT 10 0 DO I . LOOP ;

Such "colon definition" *words* are compiled on entry and added to the *dictionary*. Thus, COUNT can be executed at any time, either directly or in a program, with the result that 0 to 9 are printed on the display screen. Once new *words* are compiled, they can be removed from the *dictionary* by using the *word* FORGET in the following fashion:

FORGET wordname

as in

FORGET COUNT

Variables and Constants

Forth systems typically contain words for assigning numerical variables and constants.

CONSTANT wordname n1

creates a constant with the value of n1. When the CONSTANT is called, its numerical value is placed on the stack.

VARIABLE wordname n1

creates a variable with the value of n1. However, calling a VARIABLE by using its word name returns the *address* of that variable. The *word* @ must then be used to return the value of the named VARIABLE. VARIABLEs were apparently designed to work this way because it was assumed that program variables would be changed more often and called less frequently than true *constants*.

Forth Memory Manipulations and Assembler/Machine Coding

Like Basic, Forth can directly read and write to memory locations. The Forth equivalent of PEEK could be considered the previously mentioned @, which goes to the address

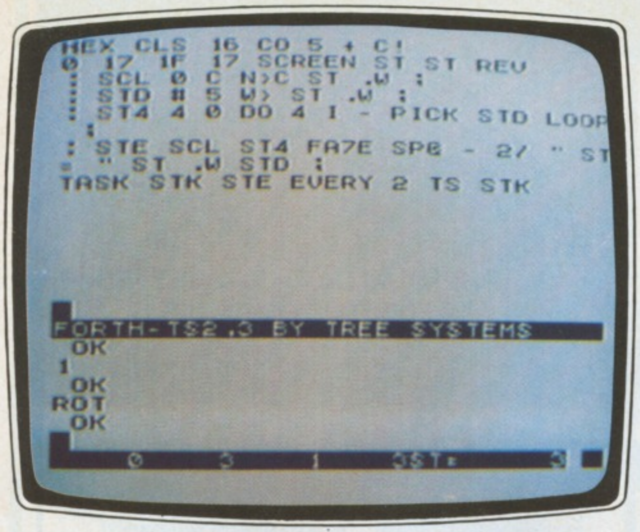


Photo 2. Split displayscreen from Pluriforth. Top block shows Forth program which makes reverse video "stack display" at the bottom. Numbers and words between the inverse lines are directory runs and display.

given at the top of the stack and loads the number at that location onto the stack. This is a 16 bit operation, so in 8 bit systems, Forth reads the least significant byte (LSB) from the supplied address and the most significant byte (MSB) from the next location. To do 8 bit reads of memory contents, the *word* C@ is used, the "C" designating a "character-size" (ASCII character codes are seven bit) read. The remaining digits of the MSB on the stack are filled out with 0's.

The FORTH equivalent of POKE is ! used as follows

addr n1!

Like @, ! uses 16 bit numbers, breaking them into most significant and least significant bytes for 8 bit machines. For writing 8 bit numbers to memory locations, the *word* C! is used. D! and D@ provide the equivalent operations for double precision (32 bit) numbers.

Forth usually includes some provisions for creating new *words* incorporating assembly language mnemonics or machine code native to the central processor used in the host computer system. The format is usually that of a *word* definition, with the wordname followed by a *word* signifying the beginning of a string of code or mnemonics. Some other *word* followed by ; indicates the end of the newly defined *word* of assembler or machine code. Thereafter, the new *word* can be executed like any other *word*.

Forth Programs

Forth programs are constructed using an *editor* peculiar to Forth systems. Programs are made up of Forth lines arrayed in *blocks* of 16 lines 64 characters in length. Each line represents a number of stack, mathematical, or logical operations, or the definition of a *word* or variable. The Forth editor usually allows the user to write or change one line of a block at a time.

An effective program uses successive lines of multiple successive blocks to build operations or functions with successively more complex *words*. Program blocks (also known as SCREENs) are usually numbered, stored in order, and retrieved by number. The final lines of a set of program blocks define a *word* which, after all program blocks are compiled, performs the desired operation when entered from the keyboard. One can see that a Forth program really represents a sort of "new language" developed by the programmer to perform a specific function.

Since Forth systems frequently use floppy disc systems for data and program storage, the use of program *blocks* similar in size to disc data blocks (16 $\times$ 64 bytes) is very convenient. However, unlike other languages, Forth can be easily used with cassette-based systems which divide tape storage space into blocks of designated length.

Other Forth Functions

Several other important constructs should be mentioned.

FENCE protects a *word* from FORGET.

DOES> is a complicated construct which allows the user to define new types of *words* with patterns of execution different from those available in the kernel. Comments can be placed in programs (and will not be compiled) if enclosed by the FORTH *word* delimiters (and).

The equivalent of Basic IF-THEN logical branches can be obtained with Forth IF *words* & numbers THEN. A more complex conditional statement can be obtained with IF *words* & numbers ELSE *words* & numbers THEN. In this construct, the code between IF and ELSE is executed if the top stack entry is non-zero. If the top stack entry is zero, the code between ELSE and THEN is executed.

AND, OR, and XOR perform logical bitwise AND, OR, or exclusive OR on the top two 16 bit stack numbers.

EMIT prints on the video display an ASCII character specified by the number on top of the stack.

For a more complete description of these and other FORTH functions, consult *Starting FORTH*.

The Pluri-Forth Operating System for Sinclair/Timex Computers

Tree Systems of Grand Rapids, Michigan, has developed a Forth operating system for the ZX81 and TS1000 computers. The system is supplied as "firmware"—a programmed 2764 8K EPROM mounted on a small circuit board. The circuit board is intended to fit into the Sinclair/Timex ROM socket after the Sinclair Basic ROM is removed. The Tree

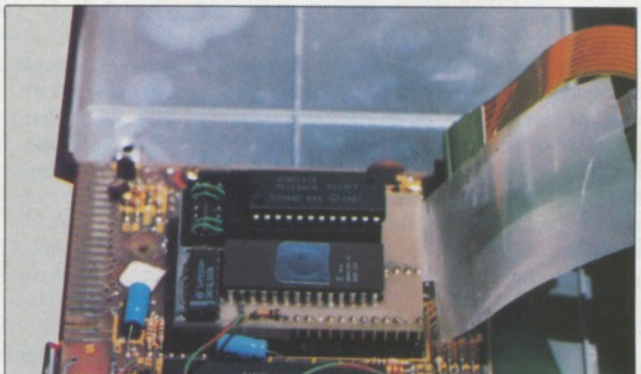


Photo 3. Tree Systems Pluriforth Eprom Board in place in the ZX81 ROM socket. The ZX81 ROM is in the additional socket next to the masked 2764 green/red Forth/Basic switch which goes to the channel 2/3 switch.

PLURI-FORTH provides some unusual features which may not be found in other Forth implementations or in any language available for the ZX/TS computers.

systems circuit board also contains a connector section for mounting the removed the Basic EPROM. Using a single control wire connected to the center post of the channel $^3/_4$ switch (or another appropriately connected switch) the user can switch at will between the Basic and Forth ROMs.

This arrangement is very powerful for several reasons. First, it is not uncommon to "crash" a Forth system by overwriting parts of memory. Having the language permanently resident in PROM means that frequently reloading the language from cassette is avoided.

Furthermore, the user can conveniently switch back and forth between Basic and Forth, which is a real advantage in testing and developing new software for previously available devices (such as the Timex/Sinclair printer or interfaces).

Finally, Forth is implemented as the language and an operating system, which provides some speed advantage over Forth implementations which "run over" and make use of the Basic operating system/monitor. The PLURI-FORTH EPROM and circuit board are shown in Photo 3.

In addition to providing a substitute operating system for the ZX81 or TS1000, PLURI-FORTH provides some unusual features which might not be found in other Forth implementations, or in any other language available for Sinclair/Timex computers.

The most outstanding of these is a true *multitasking* facility. Although the fundamental Forth "standards" include provisions for *words* defining the execution of multiple tasks, most implementations of the language for personal computers do not permit multitasking.

When I first heard about PLURI-FORTH (originally called "MULTI-FORTH), I found it very difficult to believe that true multitasking could be obtained with a system containing as little user RAM as the ZX81. Having tested the PLURI-FORTH EPROM, I can report that the ZX81 is indeed capable of concurrently running several assigned "tasks" at the same time.

The format for setting up tasks is a follows:

TASK taskname taskword/s EVERY n timeinterval taskname

for example:

TASK COUNTUP COUNT EVERY 2 TM COUNTUP

This task executes the *word* COUNT (as defined above) every 2 minutes.

PLURI-FORTH defines "timeinterval" *words* which set tasking intervals from $^1/_{60}$th of a second to 1 *YEAR!* In the task above, TM is the time interval *word* for minutes. The final COUNTUP *word* starts the execution of the task defined by the preceding *words*, and one can set up tasks with delayed execution by delaying entering the taskname second time after initially defining the task.

Photo 2 (top) illustrates the power of this multitasking function in setting up a task (called STK) which prints the stack contents on the display screen every 2 seconds. The screen shows the result of running this task, and entering numbers for stack operations. Stack numbers are displayed in Forth convention (stack top at the right) in the reverse video line at the bottom of the screen. Because this is a true multitasking operation, this stack display runs continuously, yet the user can directly execute other program lines or set up additional tasks for execution at the same time. To stop initiation and set up a new task with the same wordname, simply FORGET the TASKname.

In addition to multitasking, PLURI-FORTH provides several extremely useful extensions to Forth "standards," as well as some fundamental improvements in Sinclair/Timex hardware function. Under PLURI-FORTH, each keyboard key repeats if "held down" and the video display cursor is a rapidly blinking block, like those found in more expensive computer terminals.

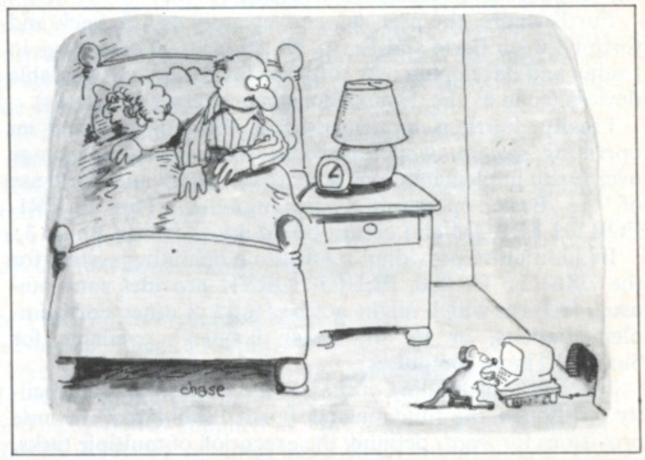
In SLOW (PLURI-FORTH has FAST and SLOW modes like standard Basic) printing to the screen is several times faster than in Basic SLOW, and the screen automatically scrolls under most conditions. This difference in screen printing is incredible: the improvement may be difficult to appreciate until one uses the *word* VLIST, which lists on the video display all of the *words* currently in the dictionary. The *words* scroll by on the screen almost too fast to read in SLOW mode!

In addition to SLOW, which constantly updates the display file, PLURI-FORTH can switch off the display during program execution using FAST (as in Basic).

In addition to these familiar models, PLURI-FORTH also has the *word* AUTO, which, when executed, turns the display off only if the execution of a *word,* task, or program takes more than 4 seconds. This is a very useful compromise between FAST and SLOW operations so familiar to Sinclair/Timex users.

PLURI-FORTH also supports two different types of editing "screen." When the system is first started, a "command" screen is displayed with a blinking cursor near the top of the display screen. Any directly executed input or word definitions cause the cursor to move down a line. When the "command" cursor reaches the bottom of the page, each additional line of input will cause the screen to scroll upward automatically, and the cursor will remain on the last line of the screen.

If <SHIFT> <EDIT> keys are depressed simultaneously, the second type of *split* screen is displayed. The top 16 lines of the display are set aside for creating and *editing* Forth programs, which are conventionally composed of blocks or "SCREENs" of 16 64-character lines, as pre-



A split screen feature allows editing along with the direct entry and execution of words.

viously discussed. A reverse graphic line delineates the bottom of the *editor* (ED) screen, below which the remaining lines constituting a *command* (CO) screen used for direct entry and execution of *words*. The CO screen works like the "whole field" screen which first appears on system *power-up*, in that program lines are directly interpreted or compiled and the screen automatically scrolls following each execution. When the ED/CO split screen is displayed, *either* the CO *or* the ED is active at one time, as indicated by the presence of an active blinking cursor located in the appropriate part of the display screen.

In addition to this unusual split-screen editor, two other related functions add to PLURI-FORTH's versatility. One of these is that the ED screen can be saved on tape as a named file, which is how Forth programs are stored. In order to deal with a program longer than a single block (or SCREEN), named blocks are saved in sequence, then reloaded and compiled in sequence.

PLURI-FORTH has the facility for automatic loading and compiling of stored program blocks (SCREENs). This function is vital for a Forth system which must make use of limited random access storage space for programs. PLURI-FORTH includes a non-vital but extremely useful extension of this capacity in that it can save named SCREENs which contain data or text. In this way, the ED screen can be used for creating data or character files.

In Pluri-Forth, most of the major Forth *words* have been used, including ordinary mathematical and stack operators, memory manipulations, base changes, loops, and *word* definitions. As previously mentioned, PLURI-FORTH uses a "non-standard" editing screen setup, so the usual FORTH *words* for displaying program blocks and editing lines are not used. Because PLURI-FORTH LOADs from and SAVEs on cassette, *words* have been included for tape operations.

PLURI-FORTH also allows text, data, and display screens to be stored on cassette using the ED screen and tape handling *words*. This kind of feature is not available with Sinclair Basic, and the user will find a whole new type of function opened up with PLURI-FORTH.

Like other Forth implementations, only integer mathematic functions (8, 16, 32, and 64 bit) are supplied, but the user can create his own floating point extension words, as described above. PLURI-FORTH also supports the convention of naming variables or constants, except constants are called INTEGERs.

PLURI-FORTH supports defining new *words* in machine code. It does so as follows:

: wordname CODE hex code numbers ;C ;

where ;C indicates the end of the sequence of hexadecimal code numbers. To assist users with constructing machine code routines, Tree Systems includes a *Zilog Programmers Reference Guide* which lists all of the Z80 operation codes by category.

The PLURI-FORTH EPROM and circuit board are also supplied with a comprehensive 104 page users manual which not only provides the user with a guide to specific operations but serves as a good general introduction to Forth for the unfamiliar.

Overall, I found PLURI-FORTH to be an excellent implementation of Forth for Sinclair/Timex computers, and it adds a number of new functions and increased speed. It provides the unheard of capacity for true *multitasking* in an ex-

PLURI-FORTH provides the unheard of capacity for true multitasking in an extremely small package which makes it easy to control peripherals.

tremely small package. I have used PLURI-FORTH in conjunction with the Computer Continum analog interface for A/D and D/A conversion, and the multitasking facility makes it very easy to set up and control sampling or operation of other peripheral devices operating on the Sinclair/Timex bus.

```
  5 FAST
 10 FOR A=30722 TO 31722
 20 POKE A,PEEK 15377
 30 NEXT A
 35 SLOW
 37 PRINT AT 0,0;"DISPLAY WHICH
BLOCK? (0-15;>15 EXIT)"
 38 INPUT N
 39 CLS
 60 IF N>=16 THEN STOP
140 FOR B=N*64+30722 TO N*64+30
783
150 PLOT B-(30721+N*64),10+(PEE
K B)/20
160 NEXT B
161 FOR X=0 TO 31
162 PRINT AT 21,X;"."
163 NEXT X
164 FOR Y=0 TO 21
165 PRINT AT Y,0;"."
166 NEXT Y
170 PRINT AT 0,0;"SAVE SCREEN?
ENTER Y OR N"
180 INPUT C$
185 PRINT AT 0,0;"SCREEN ";N;"
                     ",
190 IF C$="N" THEN GOTO 37
195 IF C$="Y" THEN COPY
200 GOTO 37
250 FOR A=30722 TO 31722
270 LPRINT AT 0,10+(PEEK A)/20;
"*"
290 NEXT A
```

Tree Systems offers to "burn" custom *words* into the EPROM at a nominal charge, allowing additional resident functions to be added/substituted into the system. Tree Systems currently offers a version of PLURI-FORTH containing printer *words* allowing one to build output display functions for the Timex/Sinclair 2040 printer. The EPROM requires a minimum of 2K RAM in any version to operate properly.

In future articles we will illustrate how FORTH can be used for a number of practical tasks. In particular, we will focus on developing control programs for peripheral devices, such as the Computer Continuum 8 channel analog-to-digital/digital-to-analog interface board, with comparisons between Forth and Basic to illustrate the function and power of the languages.

Selected References

Auer, Lawrence. "INTERP—The Kernel of Interactive Nuts." *SYNC* 3:1, pp. 43-47.

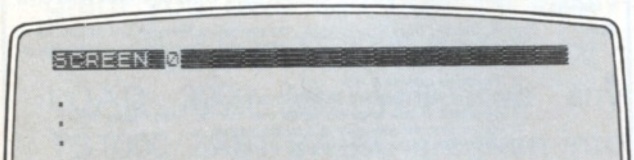
Brodie, Leo. *Starting Forth: An Introduction to the FORTH Language and Operating System for Beginners and Professionals.* Englewood Cliffs, NJ: Prentice Hall, 1981. Published by FORTH, Inc., the company responsible for building and distributing Forth; particularly useful as a guide to understanding the syntax and programming techniques.

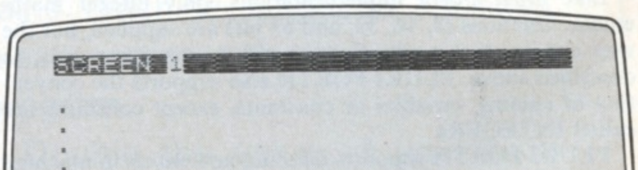
BYTE, August 1980. Special section on Forth, including: James, John S. "What is Forth? A Tutorial Introduction." *BYTE*, August 1980, pp. 100-26.

Moore, Charles. "The Evolution of FORTH, an Unusual Language." *BYTE*, August 1980, pp. 76-92.

ZILOG, Inc. *Zilog Z80 CPU Programmers Reference Guide.* Campbell, CA: Zilog, 1982.

Sample Runs.





Programming Boule

George T. Milonas

The Challenge

Donald D. Spencer concludes his *Game Playing with Basic* with a chapter of programming challenges. Among the old favorites he describes is Boule, the casino game popular in Europe and the Caribbean. This article is my response to his challenge.

The Game

The game centers on a wheel like roulette, but, unlike roulette, the wheel is stationary. A ball is spun along the raceway around the rim. As it loses momentum, it drops into one of the 18 numbered slots on the wheel. This number then is the winning number. There are only nine numbers on the wheel (1 to 9), and they appear on opposite sides of the wheel. Five is the house number with the players losing their bets when the ball drops into that slot.

George T. Milonas, 8130-H Bridgeport Way SW, Tacoma, WA 98499.

The betting combinations are what make this game so interesting. Figure 1 represents the display after a bet of $50 has been entered and after the betting combination has been selected. The player has six combinations plus the option of betting on a single number. All payoffs are even money except bets on single numbers which pay 7 to 1.

The Program Response

The program developed below to transfer Boule to the computer is straight-forward. The difficult part may be realizing that a black background is used.

The basic game principle in Boule is guessing a random number chosen by flipping the ball along the raceway. In the program we simulate this by using the computer's ability to select random numbers. However, watching the ball spin, slow down, and then drop into the slot is a big part of the excitement of Boule. The computer does not give us anything interesting to watch while it is selecting the random number that will be the winner. So to provide a little visual interest, we use a "flasher" (lines 470-500) to replace the ball and add some suspense to the game. Without the flasher or some similar delay, the winning number would appear immediately.

To provide further visual interest as well as keep track of what is going on in the game play, the Boule table layout and the betting options remain on the screen at all times. Lines 13 through 22 of the display are used for all other information and entries.

Once the program has been entered, hit RUN and ENTER. The display provides the necessary prompts to the player at each step of the way.

The program has a total of 3689 bytes as determined by using
PEEK 16396+256*PEEK 16397-16509
This total could be reduced by over 550 by assigning numbers 1, 13, 16, 18, 750, and 1300 as variables since each of these numbers appears at least 5 times in the

Figure 1.

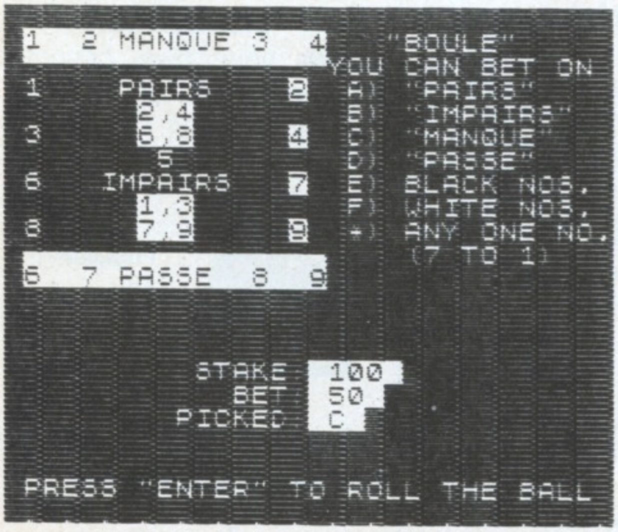
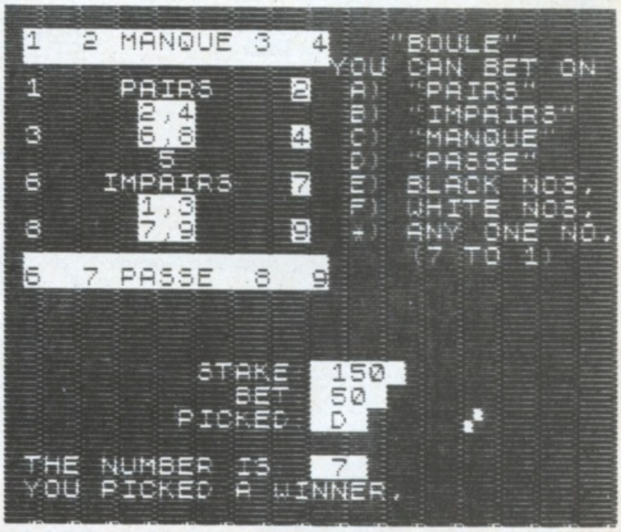


Figure 2.



As a substitute for the ball, we use a "flasher" to provide some suspense and visual interest.

program. Number 1 appears 26 times while 18 occurs 11 times. A total of 579 bytes were saved when I went through that exercise. However, since the numbers make reading the program listing easier, they are used.

Line Notes:

The notes below explain the main features of the program, give the graphics directions, and show in capital letters the text PRINTed as inverse on the screen display (only the inverse text in a particular line is given).

70: Inverse space (32); provides the black backdrop.

90: "BOULE"

100: Graphic on 6 (17).

110: Inverse 1, inverse spaces (4), PAIRS, inverse space (4).

130: Inverse 3.

140: Inverse 5.

150: Inverse 6, inverse space (3), IMPAIRS, inverse space (3).

170: Inverse 8.

180: Inverse graphic 7 (17).

190: Completes layout of Boule board.

200-270: Betting instructios.

200: YOU CAN BET ON.

210: A), inverse space, "PAIRS".

220: B), inverse space, "IMPAIRS"

230: C), inverse space, "MANQUE"

240: D), inverse space, "PASSE"

250: E), inverse space, BLACK NOS."

260: F), inverse space, WHITE NOS."

270: *), inverse space, ANY ONE NO.; "7 TO 1"

280: ALL PAYOFFS ARE EVEN MONEY--, inverse space (4), EXCEPT, inverse space, "*", inverse space, WHICH PAYS 7 TO 1.

290: NO., IS THE HOUSE NO. YOU, inverse space (5), LOSE IF, COMES UP.

300: PRESS ENTER TO CONTINUE.

350: YOUR STAKE IS, ENTER YOUR BET. From this point on the printing is on lines 13-21. The Boule board and betting instructions remain on the screen at all times.

365: Inverse space (10).

370: Inverse space (19).

380: ENTER YOUR CHOICE, (A-F) OR A NUMBER.

400: Inverse space (17); inverse space (17).

405: STAKE.

410: PICKED: Your bet and selection are on line 14 of the screen.

420: PRESS "ENTER" TO ROLL THE BALL.

440: Inverse space (32).

460: Picks the random number 1-9 as the winner.

480: Graphic on T. With 490, the attention getter.

490: Graphic on Y.

510: THE NUMBER IS:

540: SORRY ABOUT THAT HOUSE NUMBER.

580: U$ can contain either a letter or a number, depending on the betting; it is necessary to separate these two elements to determine a win or loss.

600-675: Combining here is possible, e.g., 600 and 620: IF K=1 OR K=3 AND (U$="B" OR U$="C" OR U$="E") THEN GOTO 750. For illustration the lines were left as is.

675: YOU LOSE.

750: YOU PICKED A WINNER.

800: SORRY, MY FRIEND, BUT IT SEEMS YOU ARE OUT OF CASH. One way of closing out the game when the player loses his $100 stake.

840: BOULE.

900: Changes the alpha to numeric.

920: A BONANZA FOR YOU, inverse space (2).

1000: GOOD TRY..IT COST YOU.

3000: Subroutine to erase 13-21.

3020: Inverse space (32).

3050: Delay subroutine for suspense and reading time.

```
   5 REM "BOULE"
   6 REM GEORGE T. MILONAS
   7 REM COPYRIGHT 1982
  60 FOR I=0 TO 21
  70 PRINT "                                "
  80 NEXT I
  90 PRINT AT 1,1;"1  2 MANQUE 3 4";TAB 20;" BOULE "
 100 PRINT TAB 1;"                  "
 110 PRINT TAB 1;"1    PAIRS     2"
 120 PRINT TAB 7;"2,4"
 130 PRINT TAB 1;"3";TAB 7;"6,8";TAB 15;"4"
 140 PRINT TAB 8;"5"
 150 PRINT TAB 1;"6   IMPAIRS   7"
 160 PRINT TAB 7;"1,3"
 170 PRINT TAB 1;"8";TAB 7;"7,9";TAB 15;"9"
 180 PRINT TAB 1;"                  "
 190 PRINT TAB 1;"6  7 PASSE  8 8"
 200 PRINT AT 2,17;"YOU CAN BET ON"
 210 PRINT TAB 18;"A) "PAIRS""
 220 PRINT TAB 18;"B) "IMPAIRS""
 230 PRINT TAB 18;"C) "MANQUE""
 240 PRINT TAB 18;"D) "PASSE""
 250 PRINT TAB 18;"E) BLACK NOS."
 260 PRINT TAB 18;"F) WHITE NOS."
 270 PRINT TAB 18;"*) ANY ONE NO.";TAB 21;"(7 TO 1)"
 280 PRINT AT 13,1;"ALL PAYOFFS ARE EVEN MONEY--    EXCEPT "*" WHICH PAYS 7 TO 1."
 290 PRINT AT 15,1;"NO. 5 IS THE HOUSE NO. YOU      LOSE IF 5 COMES UP."
 300 PRINT AT 20,1;"PRESS ENTER TO CONTINUE."
 310 INPUT Z$
 320 GOSUB 3000
 330 LET M=100
 350 PRINT AT 13,1;"YOUR STAKE IS $";M;" ";AT 16,1;"ENTER YOUR BET."
 360 INPUT H
 362 IF H>M THEN GOTO 360
 365 PRINT AT 16,1;"          "
 366 PRINT AT 16,16;" ";H;" "
 370 PRINT AT 13,1;"                   "
 380 PRINT AT 13,1;"ENTER YOUR CHOICE";TAB 1;"(A-F) OR A NUMBER"
 390 INPUT U$
 400 PRINT AT 13,1;"                 ";TAB 1;"                 "
 405 PRINT AT 15,10;"STAKE: ";M;" "
 410 PRINT AT 17,9;"PICKED: ";U$;" "
 420 PRINT AT 20,1;"PRESS "ENTER" TO ROLL THE BALL"
 430 IF INKEY$="" THEN GOTO 430
 440 PRINT AT 20,0;"                                "
 450 RAND
 460 LET K=INT (RND*9)+1
 470 FOR N=1 TO 50
 480 PRINT AT 17,24;"▝"
 490 PRINT AT 17,24;"▖"
 500 NEXT N
 510 PRINT AT 19,1;"THE NUMBER IS: ";K;" "
 520 IF K<>5 THEN GOTO 580
 540 IF K=5 THEN PRINT AT 13,1;"SORRY ABOUT THAT HOUSE NUMBER"
 550 LET M=M-H
 560 IF M<=0 THEN GOTO 800
 570 GOTO 1300
 580 IF U$>"0" AND U$<"A" THEN GOTO 900
 600 IF K=1 AND (U$="B" OR U$="C" OR U$="F") THEN GOTO 750
 610 IF K=2 AND (U$="A" OR U$="C" OR U$="F") THEN GOTO 750
 620 IF K=3 AND (U$="B" OR U$="C" OR U$="E") THEN GOTO 750
 630 IF K=4 AND (U$="A" OR U$="C" OR U$="F") THEN GOTO 750
 640 IF K=6 AND (U$="A" OR U$="D" OR U$="E") THEN GOTO 750
 650 IF K=7 AND (U$="B" OR U$="D" OR U$="F") THEN GOTO 750
 660 IF K=8 AND (U$="A" OR U$="D" OR U$="E") THEN GOTO 750
 670 IF K=9 AND (U$="B" OR U$="D" OR U$="F") THEN GOTO 750
 675 PRINT AT 20,1;" SORRY YOU LOSE $";H;" "
 680 LET M=M-H
 690 IF M<=0 THEN GOTO 800
 700 GOTO 1300
 750 PRINT AT 20,1;"YOU PICKED A WINNER."
 760 LET M=M+H
 770 GOTO 1300
 800 PRINT AT 13,1;"SORRY, MY FRIEND, BUT IT SEEMS  YOU ARE OUT OF CASH. ++++++++++ "
 810 GOSUB 3050
 830 CLS
 840 PRINT " BOULE  BOULE ";
 850 GOTO 840
 900 LET P=VAL U$
 910 IF K<>P THEN GOTO 1000
 920 IF K=P THEN PRINT AT 20,1;"A BONANZA FOR YOU: $";H*7;" "
 930 LET M=M+(H*7)
 940 GOTO 1300
1000 PRINT AT 20,1;"GOOD TRY..IT COST YOU $";H;" "
1010 LET M=M-H
1020 IF M<=0 THEN GOTO 800
1030 GOTO 1300
1300 GOSUB 3050
1310 GOSUB 3000
1320 GOTO 350
3000 PRINT AT 13,0;
3010 FOR I=1 TO 8
3020 PRINT "                                "
3030 NEXT I
3040 RETURN
3050 FOR I=1 TO 75
3060 NEXT I
3070 RETURN
```

Software Review

Go-fer *Paul B. Caley*

Now that disk drives are available for the ZX/TS computers (see *SYNC* 3:6 for my review of the Aerco Disk System), software in disk format is sure to follow. *Go-fer* is the first we have seen.

The tape version of *Go-Fer* can be directly transferred to disk. So you do not lose any data entry time in the upgrade.

It is a fast, flexible data handling system. Records can be selectively retrieved, displayed, or printed by either a TS2040 or a sophisticated dot-matrix printer. This powerful tool can be shaped into files such as a telephone directory, membership file, appointment calendar, library index, or label maker.

What You Get

The documentation can best be described as "handsome." It is an 8½ × 11 inch instruction booklet of 28 pages bound in an attractive glossy cover. The program is supplied on 5¼ inch floppy disk. You also get the telephone number for Speedware! Help is there if you get stuck with something or if you need advice in setting up a data base.

Hardware Required

The program requires at least 48K RAM plus the Aerco disk interface with at least one disk drive. Disk A contains the Aerco DOS and the *Go-fer* program. See Figure 1 for the first screen that comes up when you boot the disk. From this menu the *Go-fer* program is loaded. Disk B is now placed into your drive as your data disk. One disk provides 166,880 bites on one side.

The Program

Go-fer combines features of an open-ended heirarchal information system and a simple relational data base with compressed data storage. The program comes up running with the main menu displayed. The user has complete liberty to set up files tailored to his individual needs. Each record can contain as many as 253 characters which can spread over several lines. Data entry is fast and easy because previously created headings automatically appear on the screen as prompts for the next entry. If a particular heading does not apply to a record where data is being entered, the user can skip that particular heading.

Paul B. Caley, RD 1, Box 56, Duanesburg, NY 12056.

> **SYNC**
>
> **SOFTWARE PROFILE**
>
> **Name:** *Go-fer* II, version X
>
> **Type:** Database
>
> **System:** 8K ROM; 48K or 64K RAM with Aerco disk drive system
>
> **Format:** 5¼" floppy disk; cassette
>
> **Summary:** A professional information storage and retrieval system for the serious computer user in the office or home.
>
> **Price:** $100. Visa/MC.
>
> **Manufacturer:**
> Speedware
> PO Box 19138
> Austin, TX 78760

```
WELCOME TO SADOS+ 16K 8/83

C  COPY DISC (REQUIRES 2 DRIVES)
D  DISC UTILITIES
E  ENTER NAMES IN DIRECTORY
F  FORMAT DISC IN DRIVE A
H  HEX MONITOR
L  LOAD MEMORY FROM DISC
Q  QUIT
S  SAVE MEMORY ON DISC
```

The S-Search option calls an entirely new menu to the screen. The F-Form option in this menu allows the user to select or reject particular data for presentation to the video screen or printer. The search routine provides search by fields (headings) and by text scan (string search by which data is retrieved according to the information stored within any field). The logical operators AND, OR, NOT provide precise selection within a field. The TEXT SCAN and logical operator features are powerful and useful tools not found in many databases. Other approaches to searching are fully described in the manual. Helpful examples are provided for each function so the novice knows what to do and what is happening!

Print Out

An important concern in working with a database is hard copy. The program supports the use of the TS printers as well as sophisticated dot-matrix printers which allow you to send printer-specific code commands to the printer. You can choose type size, style, underlining, and the setting of line spacing during print-out. You can print out your data in formats of your choosing, e.g., a report, an invoice, a catalog list, or a membership directory.

The *Go-fer* data system also provides complete facilities for E-Editing a data base once it is created. Data may be deleted, replaced, and inserted.

Conclusion

Go-fer solved three problems I had with the database I was previously using. First, *Go-fer* allows entering large chunks of data in each record—up to 253 characters. The amount of space for data is not set aside in advance so fields can be expanded as needed. This is extremely handy as well as space saving.

Second, *Go-fer* "prints out the memory." Some information systems allow printing out only the screen. With a large mailing list, e.g., you spend most of time bringing up new screens for printing your labels. Not so with *Go-fer.* Start the printer, and in a flash you have your

```
DIRECTORY

1. AERCO DOS  16K
   USE WITH 48-64 K RAM


6. GO-FER II   RELEASE X

   SYSTEMS DISK- DISK A

   PUBLISHED BY SPEEDWARE
   PO BOX 19183
   AUSTIN, TEXAS  78760
   PHONE: 512/447-8087
```

complete list in your label format and size.

Third, *Go-fer* gives full control of a dot-matrix printer from within the program. At the same time the ZX/TS printer can be used.

This program is designed to make the best use of the Aerco disk system which many of us are using for fast, accurate data transfer. *Go-fer* is bound to make information storage and retrieval a very successful venture. ■

Software Review

The ZX Pro/File *Walt Gaby*

In *SYNC* 3:1 James Cripps found Thomas B. Woods' *ZX Data Finder* to be "the most economical program dedicated to the task of data handling" that he had seen.

While *ZX Data-Finder* is admittedly an outstanding program, it is surpassed by the *ZX Pro/File.* This new program has all the outstanding features of the earlier one plus more. It can store up to 10,890 characters in 16K RAM, yet even when filled to capacity, a given file can be brought to the screen in less than a second, according to the author's claim. Since I have no way of measuring such short time periods, I can only report my own experience of "instant" retrieval of any file I seek.

Not only is *Pro/File* fast as a result of using machine code, it is efficient. A common trait of data base programs is the presetting of file format and length. *Pro/File* accepts files of various sizes in the same program without wasting memory.

The program has three major sections:

Walt Gaby, 3325 Pierce St., San Francisco, CA 94123.

> **SYNC**
>
> **SOFTWARE PROFILE**
>
> **Name:** The ZX Profile
> **Type:** Data Handling
> **System:** 8K ROM; 16K RAM
> **Format:** Cassette
> **Summary:** Fast, efficient, versatile; excellent documentation
> **Price:** $16.95
> **Manufacturer:**
> Thomas B. Woods
> PO Box 64
> Jefferson, NH 03583

a main menu; an add/edit menu; and a file/search menu.

The MAIN MENU allows: a report on the number of bytes available; switching to the ADD/EDIT MENU and the FILE SEARCH MENU; an AUTO SEARCH option to modify the MAIN MENU to display files (or portions of files) in an orderly fashion; a PRINT FORMAT option to modify the MAIN

Hardware/Software Review

The ROMPAK System

Greg Busche

Rompak Cartridge System. Cartridge: $16.95 (kit: $11.95); EPROMs: from $9.95. Rompak Inc., Suite 100, 8206 Blackburn Ave., Los Angeles, CA 90048.

The Rompak cartridge system is an excellent low cost accessory for any ZX/TS user. The cartridge is a plugin EPROM read board with an expansion port for your RAM pack or peripherals. It also sports a neat ZIF socket.

The documentation is adequate although no schematic is supplied. The board is simple (about three parts) and easy to figure out. It is an uncased, but professionally built board that plugs solidly into the computer.

Greg Busche, 732 B Manhattan Beach Blvd., Manhattan Beach, CA 90026.

The cartridge is mapped to the normally unused 8K-16K block of memory, using an LS138 chip and, interestingly enough, a transistor. It reads not only the 2764 (8K) and 2732 (4K) EPROMs, but also, unlike other boards, the slower, cheaper 450ns ones.

A unique feature of the board is its ZIF (zero insertion force) socket. This device grips and releases the EPROM with a flip of the lever—no wrestling the EPROM in and out. No read board should be without one!

To support the board, Rompak also markets a respectable variety of software on EPROMs. Much of it is licensed from third parties. The quality is good to excellent at bargain prices. You pay cassette tape prices and get instant loading EPROMs. Instead of the usual cassette loading ritual, you need only type in a

MENU and control the format of displayed listings; a report on the current status of such format; a SAVE function.

The ADD/EDIT MENU provides for: the symbol to designate the beginning of a file; a blinking cursor to indicate the line to receive input; typing in data; deletion of individual lines or files; closing the file.

The FILE SEARCH MENU features: the command itself; display of the file on the screen; copying the displayed file to the printer; returning the display to the first file matching the COMMAND; switching back to the ADD/EDIT MENU and to the MAIN MENU. How is data in memory retrieved? After entering a SEARCH command, the FILE SEARCH MENU is displayed, and it leads you through the process.

The program is designed to relate to the Memotech Centronics interface. However, it will work with the TS2040 printer by a minor program modification.

The manual that accompanies the program could serve as a model for all who write documentation. It is understandable, well-organized, and nicely printed and bound. It has four major parts: 1) the use of the program itself; 2) an explanation of how the program works in technical, but clear detail; 3) modifications to the program; 4) two appendices: a) a complete listing of the Basic used (with all variables listed separately), and b) a complete listing of the MC used with a well-written introduction to MC for the novice.

The author has reason for being so open about the details of how his program works. He desires to provide not only an excellent product, but also educational benefits through the manual. His hope that users will re-work and refine his programs as well as tailor them for other applications is demonstrated by his willingness to share all his work through true documentation. In the belief that the *ZX Pro/File* will generate a user interest of its own, Mr. Woods has announced a quarterly newsletter entitled *Pro/File Updates* to serve as a medium of exchange for "unusual and particularly useful applications" of this powerful, but friendly new program.

---

single command and have the program run instantly.

The first EPROM I tried was an arcade game called *Timeblasters*. It ran instantly with the command RAND USR 8243 which calls the MC routine that stuffs a block of EPROM memory into the ZX/TS memory where it begins execution. Surprising speed and graphics highlighted this addictive game. After several fleets of enemy craft had been destroyed, I turned to some practical software.

Changing EPROMs in the ZIF socket was a sheer delight. *Mathpak* performs plotting of functions, integration and differentiation of functions (with replotting), curve fitting of points, and solving simultaneous equations. The program is menu driven and highly interactive. However, the plotting routine is only the standard 64 x 44 PLOT resolution. In all, a really useful program, but really nice because it is instant.

The main disadvantage of the system, though, is that it does not quite eliminate the cassette. There is no way to store a program that you have written onto the cartridge unless you possess an EPROM burner and some machine code skills. EPROMs are available with MC routines that increase the speed of SAVEing and LOADing up to 15 times. Also several companies sell EPROM toolkits, compilers, assemblers, and other utilities. Although other EPROM read boards are available, I have not found any to touch Rompak's low prices.

After using the system for about six months, I would definitely recommend the Rompak cartridge system as an inexpensive and painless way of entering the world of software on EPROMs.

resources

"Resources" lists new products and services for Sinclair and Timex/Sinclair users. Suppliers and users are invited to send brief, informative, objective descriptions of products and services along with details for ordering to: Resources, SYNC, 39 E. Hanover Ave., Morris Plains, NJ 07950.

All programs in this listing are for the TS1000/ZX81 with 16K RAM unless otherwise noted. "CC"indicates cassette format. When a supplier has more than one product listed, the name and address follow the last product.

User Groups

Florida
Timex-Sinclair User's Group
3708 Newberry Rd.
Gainesville, FL 32607

New Jersey
Timex User Group
of Central Jersey
Chris Bannister, Sec.
PO Box 267
Hopewell, NJ 08525
(609) 466-1530

Canada
ZX81 User Club
95 Wayne Ave.
Scarborough, Ont.
Canada M1R 1Y6

Victoria Sync Association
c/o Mr. Dee Shooligin
942 Cloverdale Ave.
Victoria, B.C.
Canada V8X 2T6

West Germany
ZX-Club Deutschland und
Umgebung
Postfach 967
D-7000 Stuttgart 1
West Germany

User Groups Forming

Kentucky
Louisville area.
Curtis Edwards
506 Production Ct.
Louisville, KY 40299.

Nevada
Southern Nevada area
(Las Vegas, Henderson,
Boulder City)
John Sumpelec, Jr.
2405 Howard Dr.
Las Vegas, NV 89104

Books

Timelost. By Joseph C. Giarratano et al. Que Corp. 131 pp. For 2K TS1000.

6 lost in time adventures told in comic book form and played out with a computer action game: the Mystery of Stonehenge; the Slime Creatures; the Pitdemons; the Cavern Monsters; the Dungeon of the Lord-Wizard; Cave-in. Program listings with notes.

Timex Sinclair 1000/ZX81 User's Handbook. By Trevor J. Terrell & Robert J. Simpson. Howard W. Sams & Co. 160 pp. $5.95.

Getting started; using the keyboard; number systems; strings & arrays; flow charting; logical operators; graphics; programs; how the computer works; machine code; Z80 instruction set; glossary.

The Timex Sinclair 2068. By Roger Valentine. John Wiley & Sons. 118 pp. $12.95.

50 programs. Character code; attribute table; sound; clock; calculator; simple word processor; invoicing sales ledger; using VARS; renumbering; games; patterns; sorting; locating records; graphics.

Using the Timex Sinclair 1000 and 1500. By Ralph M. Coletti. Wayne Green Publications. 83 pp. $9.97.

Reviews common mechanical problems and solutions, manual, and Sinclair Basic; programming techniques; home and business applications; teaching applications; scientific applications; hardware modifications.

Converting to Timex Sinclair Basic. By Stuart L. Bird. Wayne Green Publications. $14.95.

Gives ZX/TS equivalents for commands and functions found in other Basics for dealing with: string and numeric functions, loops, direct memory access, numeric/string conversions, graphics, system commands and editing, branching, data, matrix statements, display formatting, advanced logic and trig functions.

How to Market Your Timex Sinclair Software. 50 pages of detailed information on the TS market for turning your programs into profit. Topics included: market overview, packaging and pricing, market planning, advertising and selling techniques; list of resources and services. $11.95 ($14.95 Can.); Visa/MC. Softmark Associates, Dept. R-1, 210 5th Ave., New York, NY 10010.

Programming Projects for Your Timex Sinclair 1000. By Frank Wattenberg. Prentice-Hall. 100 pp. For 2K TS1000.

Tutorial book with 12 projects on programming with illustrative programs: reaction timer; road race; slot machine; speed maze; ecology game; games of chance; codes; calendars.

Programs for Your Timex Sinclair 1000. By Melbourne House. Prentice-Hall. 108 pp.

30 programs with notes: Random patterns, leap frog, tic-tac-toe, pinch, battleship; gambling games (e.g., craps); arcade games (e.g., Star Wars); utility programs (e.g., line renumbering); challenging (e.g., Mastermind).

Amateur Radio

Morse Code Translator (2K RAM). Reads Morse code through cassette port; scrolls translated message across screen; translates code off the air through earphone jack; generates code from a character string for recording or hearing through TV speaker; enter message in English and out comes Morse code. CC: $9.95. Thomson Software, PO Box 1266, Lombard, IL 60148.

Business/Household

Heating and Air Conditioning Simulation. Analyzes large buildings. Calculates: 1) annual heating and cooling consumption based on hourly weather, design, schedule inputs; 2) lighting, fans, pumps; 3) multiple zone buildings; 4) what-if situations. At least 32K RAM. CC and manual: $50. Michael R. Busman, 2662 Park Rd., Smyrna, GA 30080.

Construction II. 4 programs which calculate bill of material and construction costs from ground up and lead you through by asking for the specific data necessary for completion. On 2 cassettes: $30 pp. Rod Callahan, Rt. 1, Box 58, Miami, OK 74354.

Construction Estimater. User is led through a series of lists, charts, and schedules to establish construction cost estimate. Material, labor, and subcontract amounts. Part of Bill Ditrite's series of design and construction programs for the owner/builder and contractor. CC: $10. GO Industries, PO Box 330040, San Francisco, CA 94133. (415) 362-1424.

Payoff. Shows true cost of using credit cards and charge accounts. Enter new balances as you pay off or add to your accounts; explore various ways of paying off an account early; handles accounts with 2 different interest rates; shows month by month balance, finance charge, and required monthly payment.

Payout. Keep track of your spending by category and subcategory. Enter amount and date; entries sorted to category and date order within category. Gives totals for subcategories and categories, dates, amounts, and overall total.

CC: $12.95 pp. each. Ace Software, 2 E. Oak Ave., Moorestown, NJ 08057.

Fin Aid. Evaluate financial statements. Prints a series of questions seeking input from the balance sheet and income statement. Prints 18 financial measures/ratios ranging from mark up to ROI to collection period. Listing: $3.50 pp. SCDF, PO Box 5021, Richmond, KY 40475-5021.

Rental Property Management. Holds 165 detailed receipt/expense entries; auto-sort by date of transaction; auto prep of tax return; review by many criteria. Expandable to 64K. CC: $21.95 pp.; listing: $6.95; manual only: $4.95. P. Hale Software, 40 Hancock St., Boston, MA 02114.

Cryptographics Programs. 5 2K programs for secure communications. Illustrates degrees of message security, including unbreakable random codes. Booklet: $4.95. Crypt, 303 Meadowlark Lane, Durant, OK 74701.

Computer Camps

National Computer Camps. Coed, ages 9-18; all levels of experience from beginners up; June, July, August. For information contact: Dr. Michael Zabinski at (203) 795-9667 or write: National Computer Camps, Box 585, Orange, CT 06477.

Data Bases

Data-base. Create your data base, retrieve, search, or edit; write files on tape; read files from tape; menu selected printout; file parameters and field names user defined; handles various kinds of data, e.g., mailing list, inventory, collections. CC: $20; $1.50 s&h. R. Panwar, 2035 Kentland Dr., Houston, TX 77067. (713) 537-8453.

Screen-Calc. Electronic spreadsheet; 1008 cells; each can hold an 8 letter label and a number. The 180 functions, up to 80 characters in length, permit all arithmetic, trigonometric, and any other operations in the TS; loops; conditional statements; one step summing and averaging; replace functions at will; FIND and JUMP commands provide quick movement. Menu driven.

Screen-File. Holds and searches files containing over 17,000 data characters. Each file holds name, address, telephone, key line, and up to 6 additional data lines. Locate files by name, number code, key line word.

Banta Software. 8088 Highwood Way. Orangeville, CA 95662.

Kitchen Master. Keep track of kitchen items; change, delete, add, search for items. E.g., print out grocery list. Adapts to any inventory. CC: $8.95. Fun-Tech Software, 150 S. Hilsenbeck Rd., PO Box 63, Pierson, FL 32080.

Educational Programs

Making It to the Top. Personality inventory testing program; measures you against 6 scales related to personality qualities most needed for success today; randomly chosen questions; never same test twice; includes behavior modification exercises. CC: $19.95; Computer Ware Publishing, 234 Fifth Ave., 3rd Floor, New York, NY 10001. In Can.: $24.95; Computer Ware Publishing, 92 Ruskin St., Ottawa, Ont., Canada K1Y 4B2.

Personality Profile. Gives personality assessment after evaluating your responses to specific questions. CC: $3.

WHM Software, PO Box 5223, Roanoke, VA 24012.

States and Capitals. Learn the states, capitals, year of statehood, abbreviations, the 13 original colonies. US map. All 50 states highlighted. 8 modes: 7 quiz; 1 teaching. CC: $8.95. Fun-Tech Software, 150 S. Hilsenbeck Rd., PO Box 63, Pierson, FL 32080.

5 Educational Programs. Topics include geography, math, and science. Written in Basic with documentation. For information: TAG Software, PO Box 688, Naugatuck, CT 06770.

2K Edugram. 2K LOGO (learn basic LOGO), Math Raiders (answer big multiplication problems to shoot the ship), GPOT (geometric plotting of triangulation). $6. Adventure Club, 2016 Woodhillcrest Rd., Mobile, AL 36609.

Engineering

Plane Frame. Structural analysis program to determine movement and internal forces generated by external loadings on a two dimensional frame. Based on the stiffness method of matrix structural analysis; output formatted for screen and printer. CC, manual, sample problems: $29.95. J. B. Thomison, 1601 Westop Trail, Knoxville, TN 37423.

Graphics

25 Machine Code Programs. Assortment of programs from games to utility programs for using ZX/TS graphics better. $6.

High Resolution Graphics Program. Increases graphics resolution to 128 x 192. CC: $8.

Craig Bird, 2091 Carrick St., Victoria, B.C. Canada V8R 2M5.

Oscilloscope. Eye catching moving graphics. Creates full screen oscilloscope patterns that quickly rotate in real time and appear 3-D. Listing: $2 plus SASE. John Richard Coffey, PO Box 448, Scottsburg, IN 47170.

Health

Physical Fitness Analysis. For dieters, health education, or the weekend athlete. Calculates weight loss, pulse rate, target heart rate and ideal weight; includes tables of weight loss and calories used per activity. CC: $8. B. A. Mullkoff, 5578 Perry Town, W. Bloomfield, MI 48033.

Math/Science

MATSOL. Solve any number of simultaneous equations; uses Gaussian elimination with partial pivoting for a fast solution. CC: $9.95. J. B. Thomison, 1601 Westop Trail, Knoxville, TN 37423.

Statistical Correlation. Multiple regression program. Accepts up to 10 variables, computes, and prints to standard printer min, max, mean, SD, simple correlation matrix, multiple correlation with stepwise dropout routine, F-values, and probability. SASE for information. Woods End Laboratory, PO Box, Kimball Pond Rd., Vienna, ME 04360.

Speed of Sound Calculator. Gives speed simultaneously in ft/sec, MPH, knots with only a user input of the ambient air temperature at the desired altitude. Listing: $2. Robert J. Sabbia, 543 Iroquois Tr., Carol Stream, IL 60188.

Areas (2K RAM). 2 programs to approximate the area under a curve to either the x or y axis using Simpson's rule and the trapezoidal rule. Plug in your equation along with two constraints. Listing: $4. Stephen Zachev, 4859 Elmwood St., Muskegon, MI 49441.

Intermediate Math Pack (2K RAM). Compute primes, GCD, LCM, any base conversion, roots of functions, triangle solutions, linear regression, simultaneous equations, determinants, vector operations. CC, instructions, sample problems: $14.95 pp.

The Calculus (2K RAM). Sophisticated programs for numerical integration, differentiation, 1, 2, 3rd derivatives of function, 1st & 2nd differential equations by Runge-Kutta and Adams-Moulton methods. CC, instructions, sample problems: $14.95 pp.

Weather Predictor/Analyzer (2K RAM). Programs for predicting local, short term weather; computing relative humidity, dew point, temperature-humidity index, wind chill factor. Warning messages for heat stress, heatstroke, and frostbite. CC: $14.95 pp.

L. Auersbacher, 41 King St., Belleville, NJ 07109.

Analog measurement system (2900-Z). For general purpose measurement 14 channels of analog input; simple interface to Sinclair Basic; individually software configured channels, allowing any mixture of temperature, voltage, current, resistance, or frequency measurements; integrating A/D converter allows trade measurement accuracy and noise rejection for conversion time. Maximum resolution: 200 microvolts DC, 0.05 degrees F. English language control over all aspects of the process. Internal ROM. $89. Occam Research, 34 Washington St., PO Box 1055, Trumansburg, NY 14886.

Music/Sound

Talking TS1000. 1) Pronounces hex numbers of memory from any address; allows checking machine code or data without looking at screen; heard through TV speaker or tape port; resides at top of memory. 2) Record voice or sounds into computer memory; record and play back at different speeds. CC: $9.95.

Music Box. Stores notes into strings for each tune; adjustable tempo; playback tunes in any order; keeps menu of tune names; enter notes from table provided; comes with 5 preprogrammed tunes. CC: $9.95. Thomson Software, PO Box 1266, Lombard, IL 60148.

Programming Aids

16K MultiProgram. Explore the mechanics of your computer; block transfer operation; gain advanced programming skills; store up to 10 multiple programs; four games; learn how machine code works. CC: $10.95; $2 s&h. ZX Users Group of New York, Box 560, Wall St., New York, NY 10005.

Variable Dump. 268 byte MC program lists all variables and values in memory. Handles strings, arrays, token characters, numeric and E-format variables. Useful for debugging Basic programs. CC: $9.95; listing: $4.95. J. B. Thomison, 1601 Westop Trail, Knoxville, TN 37423.

16K Assembler. 2 pass assembler that has all the Z80 instructions plus the ZX calculator language. Makes your work easy through use of labels, variables, table, text, and simple expression evaluation. CC: $4.95. John Richard Coffey, PO Box 448, Scottsburg, IN 47170.

Word Processors

Z-Word. Word processor. Write, delete, insert, replace. Full print position control. Output to printer; 32 column; any number of lines per page. CC: $6.95. R. E. Woish, 32-F Plymouth Park, 1700 Butler Pike, Conshohocken, PA 19428.

Word Sinc II.4. New version includes standard WSII (*SYNC* 4:1), e.g., upper/lower case; repeat on all keys; full screen editing; search, replace, delete, tab. Drive all 80 column printers through standard parallel interface; use special printer functions from inside the text file, e.g., selectable line length; optional right justification. 16K: 8000 characters; 64K, 48,000 (up to 36 pp double spaced). Write for details and prices; give computer model number, RAM size, printer, interface. Exclusive US distributor: Gesang Associates, PO Box 452, Randallstown, MD 21133 (301/922-0767 after 6 pm).

Text Editor. 2 parts. Screen editor: lets you handle all 128 characters plus ability to rearrange paragraphs, sentences, etc. Text formatter: left and right margin adjust, automatic paragraph indentation, adjustable line length. 32 single keystroke commands. Basic listing: $2 plus SASE. John Richard Coffey, PO Box 448, Scottsburg, IN 47170.

Word Processor Program. Upper and lower case text on the Sinclair or TS2040 printer; programmable characters; many other features; MC; uses 5K of RAM. Craig Bird, 2091 Carrick St., Victoria, B.C. Canada V8R 2M5.

Games

Crazey Mazey (2K RAM). 4 games: Byte Bite (can you avoid the doggy's chomping jaws?); Slalom (ski your way through this ever-changing snowy scene); Labyrinth (scatter a few bits along the trail to find your way through 3 mazes); Cat & Mouse (catch your dinner before the cat catches you in this invisible maze).

Keystroke Management (2K RAM). 4 games: Musical Desks (it's more luck than skill to keep your desk when all about you are losing theirs); Corner Office (find your way through the invisible corporate maze to the corner office); Office Automation (will you survive in the brave new electronic workplace?); Go to the Top (corporate adventure game with the key to the executive washroom the reward).

CC: $14.95 each. 2-Bit Software, PO Box 2036, Del Mar, CA 92014.

Fastman (2K RAM). Move through a complex maze with food and power pills while ghosts chase you. Speed levels. Fastload. High score. All MC; arcade type. CC: $4.95; $1.50 s&h. Joystick: $14.95; $3.50 s&h. Zeta Base, 10451 Tennessee Ave., Los Angeles, CA 90064.

Football. Colorful, challenging strategy game for 1 or 2 players with both offensive and defensive plays are available for both players. Banta Software. 8088 Highwood Way. Orangeville, CA 95662.

Computer Boxing. Match the 25 all-time great heavyweight champions from Sullivan to Holmes. Was Ali better than Louis? Dempsey vs. Marciano? Profiles on each boxer make for realism. CC: $12.

Journey through the Asteroid Belt. Pilot your ship to avoid the asteroids. Points are based on your skill. Game gets harder. Ship has 3 shields. Each time you hit an asteroid a shield is destroyed and you lose manuvering power. Score bonus points by shooting asteroids, but you must avoid the debris. CC: $6.50. Tri-Soft, c/o R. A. Book, Seal Beach, CA 90740.

Casino Keno. Like the Las Vegas casinos; 80 number display board; players mark their tickets, selecting from 1 to 12 numbers; computer randomly selects and displays 20 numbers; winner payoffs displayed on command; odds used are those of the casinos; printer recommended. CC: $19.95 pp. Macshak

Software, 73-312 Ironwood St., Palm Desert, CA 92260.

Satellite War. You command a satellite tracking station with the fate of US and NATO forces relying on your skill in tracking down and destroying the enemy's killer satellites. Miss, and a major city, state, country may be obliterated. Fast pace. CC: $5.

Space Base Attack. Protect your space base from the aliens. Alien gives one last chance to crack his code. MC graphics; fast pace. CC: $8. WMJ Data Systems, 4 Butterfly Dr., Hauppauge, NY 11788.

River Assault. Pilot your fighter through waves of enemy missiles; sink ships; hit enemy fuel depots; fire up to 5 missiles at once; downward scrolling. 4 linked scenarios; 3 difficulty levels; 9 speeds; real time scoring. MC arcade type. CC: $15.

Jet Bomber. Manuever your jet over the scrolling landscape and through tunnels while you drop bombs on targets and fire missiles at enemy fighters in your path. Missiles from the ground make your mission more difficult. 5 levels. MC arcade type. CC: $12.

Robert J. Midura, 19 Merrifield St., Worcester, MA 01605.

TS2068

TS2068 Supershaper System. Create graphic characters on screen by joystick or inputting binary or decimal values. Characters displayed for editing, saved to tape, automatically loaded into data statements. Two programs and several data arrays; multiple character starter library; tutorial on character graphics; CC: $15.95. C. W. Associates, Dept. G. 419, N. Johnson St., Ada, OH 45810.

Software updates for TS2068. Teacher's Gradebook: $24.95. Address Book: $19.95. Appointments/Scheduler: $19.95. Financial Record Keeper II: $19.95. Invoice Control II: $14.95. Write for information and catalog. Markel Software, PO Box 2392, Secaucus, NJ 07094-0992.

TS2068 software mail order service. Over 50 programs (games, business, home, educational, programming aids from Timex, Softsync, Quicksilva, and others. CC-auto catalog: $2. TS Training Information Center, PO Box 15214, Red Bank, TN 37415. 1-615-877-6328.

States and Capitals. Find out how well you know the states and capitals. Self-teaching tool. Draws map of the US, shows where each state is, locates capital cities. Names must be spelled correctly. The computer will ask you for the state, the capital, or a combination. CC: $15.95; Cart.: $29.95.

Math Wizardry I. Teaches young children the processes involved in solving addition and subtraction problems. Moving displays lead the user through the step by step solution. Digits in answer entered from right to left. CC: $15.95.

Fun Golf. Up to two golfers can enjoy a 9- or 18-hole trek around the green. You must hit the ball from tee to hole and avoid the bunkers, water treads, trees, and roughs to score low. CC: $15.95.

Crazybugs. You are in a maze in which you must get the eggs laid by the bugs, put them in a box, and exit the maze. Although you are faster than the bugs, they are intelligent and will trap you. The screens increase in difficulty. At the higher levels super bugs join the chase. CC: $19.95; Cart. $29.95.

The Timex Computer Corp., Waterbury, CT 06725.

Gulpman. In Gulpland a typical Gulpman dashes through a maze of apple trees to satisfy his voracious appetite for apples, but the Wormoids are constantly after him. They move faster and faster, but he has 9 lives. 15 mazes; 9 levels; sound effects. $19.95.

Cyberzone. With your lasers you must chip away the spinning Cyber spacecraft and wipe out the notorious Cyberzone leader by hitting his only vulnerable spot: the left foot. You must avoid his shots and down his fighters. 5 levels. Lasers controlled by keys, joystick, or voice. $19.95.

Personal Accountant. 3 integrated programs. Compile reports, automatic posting, data base, add messages to entries, profit/loss statements, amortization/payment schedules; create as many accounts as you need; single entry posted into proper categories. $24.95.

Zeus Assembler. Machine code programming is made simple. Use the full Z80 mneumonics instruction set. Editor allows cursor movement in 4 directions; insert, delete, clear line and screen. Display current registers and alter them. Save MC by entering one letter commands; substitute lines of code; Move blocks around in memory. Minimonitor accesses memory. $19.95.

Softsync, Inc. 14 E. 34th St., New York, NY 10016. (212) 685-2080.

classified ads

CLASSIFIED RATES: Per Word, 15 Word Minimum $1.35. GENERAL INFORMATION: Prepayment discounts available. Payment must accompany order except credit card—Am. Ex., Diners, MC, VISA (include exp. date)—or accredited ad agency insertions. Copy subject to publisher's approval; must be typewritten or printed. First word set in caps. Advertisers using P.O. Boxes MUST supply permanent address and telephone number. Orders not acknowledged. They will appear in next available issue after receipt. Send order & remittance to: Classified Advertising, SYNC Magazine, 1 Park Avenue, New York, NY 10016. To charge your ad to a major credit card, call Lois Price at (212) 725-7226. For Customer Service, call (212) 725-4312.

SOFTWARE

TAX RETURN ORGANIZER. Program written by an accountant that helps you organize your tax records. Program asks questions which you answer by using your W-2 forms, interest statements, etc. Can be used year after year. program provides hard copy for tax records. For ZX81/TS1000 with 16K and printer. Cassette $15.00. Listing $8.00. WMJ Data Systems, 4 Butterfly drive, Hauppauge, NY 11788.

LOAN AMORTIZAION. Provide any three of the following factors: Amount, interest rate, number of payments, or payment amount. program will calculate the unknown factor. Any payment factor can be used; monthly, weekly, daily, etc. program will break out the interest and principal portions of each payment for any period desired; it will also total each column. Should be used with printer but can be used without one. For ZX81/TS1000 with 16K. Cassette $15.00. Listing $8.00. WMJ Data Systems, 4 Butterfly Drive, Hauppauge, NY 11788.

FREE Timex programs! SASE brings catalog inexpensive! Mailing lists, eZRAEZRA, Box 5222-YM San Diego, California 92105.

CANADIANS; BEST SELECTION lowest prices. FREE catalog. JDC 5-14, Sentinel Rd. Downsview, Ont. Canada M3M 2Y5. U.S. Inquiries.

SOFTWARE CHEAP! FREE special programs plus catalog. Send large SASE. ALECK, Box 493, Groves, TX 77619.

ZX81 TEXT PROCESSOR16—64K RAM, 1100-7700 words. Write, read, edit, print, save, clear text. Full or window-screen editing, $12.50 tape/instructions. Check/M.O.: A. Rodriguez, 1605 Pennsylvania Ave., 204, Miami Beach, FL 33139.

HAM/SCANNER/SWL OPERATORS. Super log book and other software for Timex, TI99 and Commodore. SASE for information. Kentronics, P.O. Box 586, Vernon, AL 35592.

TREE FORTH for the 2K+ Sinclair User—$49.95. Multitasking forth for ZX81 and TS 1000. Free information. Contact: Softmagic Corp., 1213 West High Street, Bryan, Ohio 43506. (419) 636-4531.

CHARTS AND GRAPHS package for TS/1000. Spharler Software Systems, Nine Springbrook Circle, Sacramento, California 95831.

TS/1000-Great values on home/finance software. Free information: Ace Software, 2 E. Oak Ave., Moorestown, NJ 08057.

ELIZA FOR 16K T/S. Does your computer answer you? Eliza does. Simulates psychotherapy/computer intelligence. Cassette $14.95. Sinclair Softech, P.O. Box 82 S, Sinclair, WY 82334.

MICRO-LOAD HAS UNUSUAL software for your computer. Fully guaranteed. Info: Box 1095, TorC, NM 87901.

TS 1000, ZX81, TI 99/4A, VIC-20 SOFTWARE. Send for FREE Catalog: Midwest Software, 9922 Harwich, Crestwood, MO 63126.

FREE SOFTWARE CATALOG for X-81/TS-1000 & TS 2068 users. Household Bookkeeper, Calendar Planner, Biorhythms, Blackjack, Jotto, Quizzers, Toggle, Sunrise/Sunset, and more! Special Discount! Write I.M.S. ENTERPRISES, Box 4503, Lancaster, CA 93539.

CRYSTAL COAST SOFTWARE: Educational, Entertaining and Practical Programs. Send for FREE catalog: PO Box 233, MOREHEAD, NC 28557.

COMPUTER SOFTWARE

T/S 2068 Disassembler and ROM Explorer. Uses machine code routines for fast listings to screen or printer. Decimal addresses, Zilog mnemonics, labeling of system variables, DEFB/DEFW modes, disassembly relocating, bank-switched ROM access. Also produces cross-reference listings of direct jumps and calls revealing major entry points and sub-routines. A quality tool for debugging/documenting own code or studying T/S 2068 ROM. Cassette $15.95. J.C. Kilday Associates, Central Ave., Peaks Island, ME 04108.

COMPUTER EQUIPMENT/SUPPLIES

COMPUTER PROGRAMMING AID—A 16 Bit Dec/Bin/Hex Slide Convertor. $3.75. U.S. inquiries. R Harder, 995 Shakespeare Ave., North Vancouver, BC V7K 1E7 Canada.

GAMES

SPACE BASE ATTACK. New MC game for ZX81/TS1000 w/16K. Excellent graphics and fast paced. Cassette $8.00. WMJ Data Systems, 4 Butterfly Dr. Hauppauge, NY 11788.

PARCLE *Robert R. Wertz*

Have you ever wished that you could clear just a portion of the screen? Recently, while working on a game program, I was faced with this problem. The playfield was printed on the upper portion of the screen, and I wanted to clear only the lower portion to print different prompting messages.

The PRINT AT command seemed to be the solution; however, I soon found out that, if the new message did not completely overwrite the old, some very unusual messages could appear. A nested FOR-NEXT loop could be used to generate spaces to overwrite the area before the new message is printed, but, if the area is very large, this could take several seconds. The only other solution seemed to be a machine code subroutine.

The machine code subroutine shown in Figure 1 searches the display file from a predetermined starting point, testing each location to determine if it contains the 118 end of line delimiter. If it does not, the byte is set to 0.

Figure 1. PARtial CLEaring MC subroutine.

Address	Machine Code (d)	Label	Instruction	Comments
16514	30,0		LD E,0	;set E to 0
16516	62,117		LD A,117	;set A to 118
16518	60		INC A	
16519	33,14,64		LD HL,16398	;set HL to address
16522	78		LD C,(HL)	of next PRINT
16523	35		INC HL	location in
16524	70		LD B,(HL)	display file
16525	197		PUSH BC	
16526	225		POP HL	
16527	1,74,1		LD BC,330	;set counter
16530	205,152,64		CALL,16536	;GOSUB "START"
16533	201		RETURN	;RETURN to Basic
16534	224	BC TEST	RET,PO	;if BC=0 RETURN from SUB
16535	35		INC HL	
16536	237,161	START	CPI	;test (HL) for 118
16538	43		DEC HL	
16539	40,249		JP Z,"BC TEST"	;if (HL)=118 jump to BC TEST
16541	115		LD (HL),E	
16542	24,246		JR,"BC TEST"	

NOTE: As shown in the article, this routine will clear the screen from PRINT position 12,0 to the end of the screen.

To understand the routine better, let's look at the display file and one of the system variables related to it. First, and this is very important, the manual states that, if the total amount of memory, according to the variable RAMTOP, is larger than $3^1/_4$K, the display file is padded out with 24*32 spaces and generally remains at its full size. (Caution: Should the display file become reduced in size, it is possible to overwrite the VARIABLES area of memory. See the manual for details.) At addresses 16398 and 16399 is the variable DF-CC, the address of the next PRINT position in the display file. The manual states that this variable can be POKEd to change the location of the next PRINT, but an easier method is to use the PRINT AT x,x command. This saves calculating the address since the monitor will do it for you.

Now let's look at the machine code. The first part of the program sets up the variables. The E register holds the 0 used to fill the display. In the A register is the 118 for the end of line test. The HL pair is the address of the display location to be tested, and the BC pair contains the number of locations to be tested.

Figure 2. MC loader program.

```
   1 REM XXXXXXXXXXXXXXXXXXXXXXX
XXXXXXX
  10 FOR I=16514 TO 16543
  20 SCROLL
  30 PRINT I;TAB 8;
  40 INPUT N
  50 PRINT N
  60 POKE I,N
  70 NEXT I
```

Type in this program, RUN, and ENTER. Then type in the numbers from the machine code column in Figure 1.

At this point in the program we execute a CALL to the test routine labeled START. The CPI instruction compares the contents of the A register to the contents of the address pointed to by the HL pair. If both equal 118, the zero flag is set. The HL pair is then incremented, and the BC pair is decremented. If the BC pair now equals zero, the parity flag is set for "odd parity." Since the CPI instruction incremented the HL pair, we now decrement it so that it still indicates the current location in the display file.

Next, if the zero flag was set, indicating that the location contained the end of line delimiter, we jump to BC TEST. Otherwise, the location is filled with 0, and then we jump to BC TEST. At BC TEST, if the parity flag indicates the "odd parity," we return to the instruction following the CALL which is the RETURN to Basic. If the BC pair is now zero, the HL pair is incremented and the next location is tested.

The program in Figure 1 is held in the first Basic program line as a REM. To use the program, execute the two Basic lines:

```
110 PRINT AT 12,0
110 LET X=USR 16514
```

The PRINT AT location should be the place from which you wish to start clearing. I called this routine from three different locations in my program and used the command

```
LET X=USR 16514
```

each time with no problems.

In addition to using this program for clearing part of the screen, several other possibilities exist. Since the addresses in the program will not change if it is kept in the first Basic line, the value in the E register could be POKEd to make it any character and the value in the BC pair POKEd for any number of locations, allowing you to quickly fill any portion of the screen with any combination characters.

Robert A. Wertz, 19229 Alcoy, Detroit, MI 48205

Index to Advertisers

Now...Reston has a complete line of software and books for the Timex Sinclair & ZX-81 computers.

RESTON COMPUTER GROUP
Reston Publishing Company, Inc.

SoftWare

- ☐ R1261-8 **Demolition/Ten Pin** — Two action-packed games in one package! $9.95
- ☐ R4608-7 **Monarch** — Control land! Rule People! See how long you can keep the throne! $12.95
- ☐ R1751-7 **Escape from Shazzar!** — Find the temple and escape the evil that lurks around every corner! $12.95
- ☐ R0705-5 **Casino Pak I: One-Arm Bandit and Blackjack** — Take a trip to Las Vegas! Play against your computer and see who comes out the big winner! $14.95
- ☐ R4953-7 **The Nowotnik Puzzle** — Can you unscramble the puzzle? It's harder than it looks! $14.95
- ☐ R0738-4 **Championship Chess** — Good news for chess players! Your computer challenges you to this award-winning game! $14.95
- ☐ R6966-7 **Self-Teaching Software for the Timex Sinclair 1000—Level 1** $29.95
- ☐ R6967-5 **Self-Teaching Software for the Timex Sinclair 1000—Level 2** $29.95
- ☐ R6968-3 **Self-Teaching Software for the Timex Sinclair 1000—Level 3** $29.95
- ☐ R6970-9 **Self-Teaching Software for the Timex Sinclair 1000—Level 4** $29.95
- ☐ R6971-7 **Self-Teaching Software for the Timex Sinclair 1000—Level 5** $29.95
- ☐ R6972-5 **Self-Teaching Software for the Timex Sinclair 1000—Level 6** $29.95

BookWare

- ☐ R7697-7 **The Timex Sinclair 1500/1000 Pocket Book** — A complete handbook for the most popular 'little' computer around! $9.95
- ☐ R0348-4 **BASIC Basics for the Timex Sinclair 1500/1000** — Learn BASIC and discover the power of the computer $9.95
- ☐ R1890-3 **50 1K/2K Games for the Timex Sinclair 1500/1000** — Dogfight! Ski Run! UFO Attack! Programming instructions turn your computer into a game arcade! $9.95
- ☐ R0347-6 **Basics of Timex Sinclair 1500/1000 BASIC** — A wonderful introduction to the world of BASIC! $9.95
- ☐ R2088-7 **49 Explosive Games for the Timex Sinclair 1500/1000** — Galactic Intruders, Checkers, Smuggler's Gold — Here are programming instructions for your favorite games! $9.95
- ☐ R4255-7 **Mastering Machine Code on Your Timex Sinclair 1500/1000** — Finally, a "simple English" guide to machine code! $9.95
- ☐ R4190-6 **Making the Most of Your Timex Sinclair 1500/1000** — A great way to learn about each feature and function of your computer! $9.95
- ☐ R1816-9 **Explorer's Guide to the Timex Sinclair 1500/1000** — Discover the limitless capabilities of your computer $9.95

☐ Check enclosed ☐ VISA ☐ MasterCard

Name ____________________

Address ____________________

City ____________ State ________ Zip ________

Card # ____________________

Interbank # ____________ Expires ________

Signature ____________________

For the Timex Sinclair 1000 Series and ZX-81 Computers / Available from your local bookstore OR Timex Sinclair Computer Outlet

Reston Publishing Company, Inc., Attn: Marketing, 11480 Sunset Hills Road, Reston, VA 22090 **Or call toll-free** 800-336-0338

CIRCLE 42 ON READER SERVICE CARD


THE SECRETS OF PERFECT MEMORY:
ONE AND ONE HALF EARTH DOLLARS
AT LAST: THE WHOLE TRUTH ABOUT FLOPPIES.
Amazing book reveals all!
How to keep from brainwashing your disk so it never loses it's memory.
How fingerprints can actually damage disks. Unretouched Kirlian photographs of UFO's (Unidentified Floppy Objects)! The incredible importance of making copies: the Department of Redundancy Department—and what goes on when it goes on! Powerful secret methods that scientists claim can actually prevent computer amnesia! All this, and much more . . .
In short, it's an 80-page plain-English, graphically stunning, pocket-sized definitive guide to the care and feeding of flexible disks.
For The Book, ask your nearest computer store that sells Elephant™ disks, and bring along one and one half earth dollars.
For the name of the store, ask us.
Elephant Memory Systems® Marketed exclusively by Dennison Computer Supplies, Inc., 55 Providence Highway, Norwood, MA 02062. Call toll free 1-800-343-8413. In Massachusetts, call collect (617) 769-8150. Telex 951-624.
THE SECRETS OF PERFECT MEMORY
"NEVER FORGETS"
CIRCLE 27 ON READER SERVICE CARD